capstone-report

November 20, 2018

1 Machine Learning Engineer Nanodegree

1.1 Capstone Project

Felipe Santos
October 23th, 2018

1.2 I. Definition

1.2.1 Project Overview

Optical character recognition (OCR) is the process to transform text images to text in a computer format, giving us information in an unstructured way, so is necessary to classify this data to gain knowledge about them and to have better document retrieval. The Tradeshift company make on that foundation a machine learning based product to classify the text blocks in a document to dates, address, and names to enrich the data of the OCR process. This organization resolves to host a competition on Kaggle, a data science platform, opening their data for community try to beat their machine learning model to this classification problem. The competition and the dataset can be access through this link and is available to the Kaggle community who intends to beat their benchmark like me.

1.2.2 Problem Statement

In this competition, we have to create a supervised machine learning algorithm to predict the possibility for a block of text being from a particular label, however, the block can have multiple labels. For all the documents, words are detected and combined to form text blocks that may overlap to each other. Each text block is enclosed within a spatial box, which is depicted by a red line in the sketch below. The text blocks from all documents are aggregated in a data set where each text block corresponds to one sample (row). The text is inputted by the OCR and the host organization gives us some features like the hashed content of the text, position, and size of the box, if the text can be parsed as a date, as a number and include information about the surrounds text blocks in the original document. The final classifier is intended to beat the benchmark of the Tradeshift organization, some tasks involved to reach that goal are:

- Download and preprocess the Tradeshift dataset;
- Do some feature engineering;
- Train different classifiers;
- Tuning the hyperparameters of the algorithm;
- Beat the benchmark.

1.2.3 Metrics

The evaluation metric chosen by the organizers for this competition was the negative logarithm of the likelihood function averaged over Nt test samples and K labels. As shown by the following equation a + b =c. On the equation:

$$\text{LogLoss} = \frac{1}{N_t \cdot K} \sum_{idx=1}^{N_t \cdot K} \text{LogLoss}_{idx}$$

$$= \frac{1}{N_t \cdot K} \sum_{idx=1}^{N_t \cdot K} [-y_{idx} \log(\hat{y}_{idx}) - (1 - y_{idx}) \log(1 - \hat{y}_{idx})]$$

$$= \frac{1}{N_t \cdot K} \sum_{i=1}^{N_t} \sum_{j=1}^{K} \left[-y_{ij} \log(\hat{y}_{ij}) - (1 - y_{ij}) \log(1 - \hat{y}_{ij})\right]$$

- f is the prediction model
- θ is the parameter of the model
- $\hat{y}_{ij}$ is the predicted probability of the jth-label is true for the ith-sample
- log represents the natural logarithm
- $idx = (i - 1) * K + j$

This function penalizes probabilities that are confident and wrong, in the worst case, prediction of true(1) for a false label (0) add infinity to the LogLoss function as $-log(0) = \infty$, which makes a total score infinity regardless of the others scores.

This metric is also symmetric in the sense than predicting 0.1 for a false (0) sample has the same penalty as predicting 0.9 for a positive sample (1). The value is bounded between zero and infinity, i.e. LogLoss $\in [0, \infty)$. The competition corresponds to a minimization problem where smaller metric values, LogLoss ~ 0, implies better prediction models. To avoid complication with infinity values the predictions are bounded to within the range $[10^{-15}, 1 - 10^{-15}]$

Example This is an example from the competition If the 'answer' file is:

```
id_label,pred
1_y1,1.0000
1_y2,0.0000
1_y3,0.0000
1_y4,0.0000
2_y1,0.0000
2_y2,1.0000
2_y3,0.0000
2_y4,1.0000
3_y1,0.0000
3_y2,0.0000
3_y3,1.0000
3_y4,0.0000
```

And the submission file is:

```
id_label,pred
1_y1,0.9000
1_y2,0.1000
1_y3,0.0000
1_y4,0.3000
2_y1,0.0300
2_y2,0.7000
2_y3,0.2000
2_y4,0.8500
3_y1,0.1900
3_y2,0.0000
3_y3,1.0000
3_y4,0.2700
```

the score is 0.1555 as shown by:

$$L = -\frac{1}{12}\left[log(0.9) + log(1-0.1) + log(1-0.0) + log(1-0.3) + log(1-0.03) + log(0.7) + log(1-0.2) + log(0.8\right.$$

1.3 II. Analysis

1.3.1 Data Exploration

In this section, I will analyze the training dataset of the competition, have some descriptive statistics and exploring the features trying to define the characteristics of this data.

1. Section **??**
2. Section **??**
3. Section 1.3.1
4. Section **??**
5. Section **??**

Loading Data

```
In [43]: %load_ext autoreload
         %autoreload 2

         import src.describe as d
         import src.pre_processing as pre
         import src.plots as pl
         import pandas as pd
         import numpy as np
         import gc
         import pickle
         import os.path
         from scipy import sparse
         from sklearn.decomposition import PCA
         from sklearn.model_selection import train_test_split
         from sklearn.metrics import log_loss
```

```
pd.set_option('display.max_columns', None)
pd.set_option('display.expand_frame_repr', False)
pd.set_option('max_colwidth', -1)
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

First look The dataset can be download at this link and have to be extracted at the root_folder/input folder. The objective is to open the dataset and the labels, see the firsts rows and the shape.

```
In [2]: train_features = d.read_train_features()
        train_labels = d.read_train_labels()

        train_features.shape
```

```
Out[2]: (1700000, 146)
```

```
In [3]: train_features.head()
```

```
Out[3]:    id   x1   x2                                                x3
        0  1    NO   NO   dqOiM6yBYgnVSezBRiQXs9bvOFnRqrtIoXRIElxD7g8=  GNjrXXA3SxbgD0dTRblAPO9
        1  2    NaN  NaN  NaN                                               NaN
        2  3    NO   NO   ib4VpsEsqJHzDiyL0dZLQ+xQzDPrkxE+9T3mx5fv2wI=  X6dDAI/DZOWvu0Dg6gCgRoN
        3  4    YES  NO   BfrqME7vdLw3suQp6YAT16W2piNUmpKhMzuDrVrFQ4w=  YGCdISifn4fLao/ASKdZFhG
        4  5    NO   NO   RTjsrrR8DTlJyaIP9Q3Z8s0zseqlVQTrlSe97GCWfbk=  3yK2OPj1uYDsoMgsxsjY1Fx
```

```
In [4]: train_features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1700000 entries, 0 to 1699999
Columns: 146 entries, id to x145
dtypes: float64(55), int64(31), object(60)
memory usage: 1.8+ GB
```

```
In [5]: train_labels.shape
```

```
Out[5]: (1700000, 34)
```

```
In [6]: train_labels.head()
```

```
Out[6]:    id  y1  y2  y3  y4  y5  y6  y7  y8  y9  y10  y11  y12  y13  y14  y15  y16  y17  y18
        0  1   0   0   0   0   0   0   0   0   0   0    0    0    0    0    0    0    0    0
        1  2   0   0   0   0   0   0   0   0   0   0    0    0    0    0    0    0    0    0
        2  3   0   0   1   0   0   0   0   0   0   0    0    0    0    0    0    0    0    0
        3  4   0   0   0   0   0   0   0   0   0   0    0    0    0    0    0    0    0    0
        4  5   0   0   0   0   0   0   0   0   0   0    0    0    0    0    0    0    0    0
```

```
In [7]: train_labels.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1700000 entries, 0 to 1699999
Data columns (total 34 columns):
id     int64
y1     int64
y2     int64
y3     int64
y4     int64
y5     int64
y6     int64
y7     int64
y8     int64
y9     int64
y10    int64
y11    int64
y12    int64
y13    int64
y14    int64
y15    int64
y16    int64
y17    int64
y18    int64
y19    int64
y20    int64
y21    int64
y22    int64
y23    int64
y24    int64
y25    int64
y26    int64
y27    int64
y28    int64
y29    int64
y30    int64
y31    int64
y32    int64
y33    int64
dtypes: int64(34)
memory usage: 441.0 MB
```

Metadata In this section, we will categorize the columns to try to facilitate the manipulation utilizing the information the description of the competition gives us before. We'll store: * **role**: input, id, target * **dtype**: int, float, str * **category**: content, numerical, boolean * **varname**: name of the column

```
In [3]: meta = d.create_features_meta(train_features)
        meta.head(10)
```

```
Out[3]:           role   category    dtype
        varname
        id          id     numerical  int64
        x1       input     boolean    object
        x2       input     boolean    object
        x3       input     content    object
        x4       input     content    object
        x5       input     numerical  float64
        x6       input     numerical  float64
        x7       input     numerical  float64
        x8       input     numerical  float64
        x9       input     numerical  float64
```

Extract all boolean features:

```
In [9]: meta[meta.category == 'boolean'].index
```

```
Out[9]: Index(['x1', 'x2', 'x10', 'x11', 'x12', 'x13', 'x14', 'x24', 'x25', 'x26',
               'x30', 'x31', 'x32', 'x33', 'x41', 'x42', 'x43', 'x44', 'x45', 'x55',
               'x56', 'x57', 'x62', 'x63', 'x71', 'x72', 'x73', 'x74', 'x75', 'x85',
               'x86', 'x87', 'x92', 'x93', 'x101', 'x102', 'x103', 'x104', 'x105',
               'x115', 'x116', 'x117', 'x126', 'x127', 'x128', 'x129', 'x130', 'x140',
               'x141', 'x142'],
              dtype='object', name='varname')
```

See the quantity of feature per category:

```
In [10]: pd.DataFrame({'count' : meta.groupby(['category', 'dtype'])['dtype'].size()}).reset_in
```

```
Out[10]:     category    dtype  count
          0  boolean    object   50
          1  content    object   10
          2  numerical  int64    31
          3  numerical  float64  55
```

Descriptive Statistics In this section we will apply the *describe* method on the features splited by category and dtype to calculate the mean, standart deviation, max, min...

Numerical float variables

```
In [3]: float_features = meta[(meta.category == 'numerical') & (meta.dtype == 'float64')].index
        float_train_features = train_features[float_features]
        float_train_features_describe = float_train_features.describe()
        float_train_features_describe
```

```
Out[3]:                  x5            x6            x7            x8            x9           x
        count  1.700000e+06  1.700000e+06  1.700000e+06  1.700000e+06  1.700000e+06  1.700000e+
```

```
mean   9.551493e-01  5.531406e-02  7.906443e-01  1.731225e-01  4.462953e-01  4.196774e-
std    5.278641e-01  1.318832e-01  3.549407e-01  3.326885e-01  3.026847e-01  2.945485e-
min    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00 -1.042755e+00 -5.919283e-
25%    6.367211e-01  0.000000e+00  8.438324e-01  0.000000e+00  1.961279e-01  1.670404e-
50%    1.270115e+00  0.000000e+00  9.588627e-01  0.000000e+00  4.393339e-01  4.002242e-
75%    1.414798e+00  5.837871e-02  1.000000e+00  1.451906e-01  6.866182e-01  6.822070e-
max    2.732124e+00  9.987901e-01  1.000000e+00  1.753333e+00  1.942155e+00  7.929372e-
```

```
In [12]: float_train_features_describe.loc()[['min','max']]
```

```
Out[12]:            x5       x6   x7        x8        x9       x16       x19   x20  x21       x
         min  0.000000  0.00000  0.0  0.000000 -1.042755 -0.591928 -0.352018 -46.0  0.0 -0.576
         max  2.732124  0.99879  1.0  1.753333  1.942155  7.929372  0.999786  14.0  1.0  7.968
```

```
In [13]: float_train_features.isnull().any().any()
```

```
Out[13]: False
```

The features that are scaled between [0,1] are: x6, x7, x21, x37, x38, x52, x67, x68, x82, x97, x98, x112, x122, x123, x137.

So we could apply scaling on the other features depends on the classifier.

And we don't have any NaN values on this features.

Numerical int variables

```
In [4]: int_features = meta[(meta.category == 'numerical') & (meta.dtype == 'int64') & (meta.ro
        int_train_features = train_features[int_features]
        int_train_features_describe = int_train_features.describe()
        int_train_features_describe
```

```
Out[4]:                x15           x17           x18           x22           x23           x
        count  1.700000e+06  1.700000e+06  1.700000e+06  1.700000e+06  1.700000e+06  1.700000e-
        mean   6.154404e+00  4.487084e+00  8.096322e+00  2.301595e+03  1.874765e+03  2.814097e-
        std    8.957511e+00  4.623426e+00  7.123864e+00  1.745120e+03  1.517991e+03  4.409801e-
        min    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00 -1.000000e-
        25%    1.000000e+00  2.000000e+00  3.000000e+00  1.261000e+03  8.920000e+02  0.000000e-
        50%    3.000000e+00  4.000000e+00  7.000000e+00  1.263000e+03  8.920000e+02  1.000000e-
        75%    7.000000e+00  6.000000e+00  1.100000e+01  4.400000e+03  3.307000e+03  4.000000e-
        max    1.530000e+02  2.370000e+02  2.190000e+02  1.950000e+04  1.416700e+04  6.720000e-
```

```
In [ ]: int_train_features_describe.loc()[['min','max']]
```

```
In [16]: int_train_features_describe.isnull().any().any()
```

```
Out[16]: False
```

All the int numerical features are not scaled, so depending on the algorithm we have to scale the feature, we don't have any missing value. The problem here is we don't know when the feature is a categorical feature or a quantitative.

Content variables

```
In [10]: content_features = meta[(meta.category == 'content')].index
         train_features[content_features].replace('nan', np.NaN)
         train_features[content_features] = train_features[content_features].astype(str)
         content_train_features = train_features[content_features]
         content_train_features_describe = content_train_features.describe()
         content_train_features_describe
```

Out[10]:

	x3	x4		x34	
count	1700000	1700000	1700000		1700000
unique	201882	26429	241246		35701
top	nan	nan	MZZbXga8gvaCBqWpzrh2iKdOkcsz/bG/z4BVjUnqWT0=		YvZUuCDjLu9Vv
freq	248263	248263	57978		84162

```
In [12]: # flattening all the words to count them
         all_words = pd.Series(content_train_features.values.flatten('F'))
         all_words = all_words.to_frame().reset_index()
         print('total words={}'.format(all_words.shape[0]))
         all_words = all_words.rename(columns= {0: 'words'})
         all_words = pd.DataFrame({'count' : all_words.groupby(['words'])['words'].size()}).res
         all_words.sort_values('count', ascending=False).head(10)
```

```
total words=17000000
```

Out[12]:

	words	count
790221	nan	1021642
565834	YvZUuCDjLu9VvkCdBWgARWQrvm+FSXgxpOzIrMjcLBc=	392698
538278	X6dDAI/DZOWvu0Dg6gCgRoNr2vTUz/mc4SdHTNUPS38=	356811
692301	hCXwO/JldK5zcd9ejOD1FwmEgCf96eTdEVy7OtY2Y2g=	317031
376725	MZZbXga8gvaCBqWpzrh2iKdOkcsz/bG/z4BVjUnqWT0=	273502
199214	B+EJpnEbkYtLnwDQYN1dP1rcfnoCnxAjKLYwQZE07Ew=	260233
15027	+yhSY//Hpg7u0bSA7NYmcmRFgv3bF4Tw3BMHrBqaTtA=	260166
528829	WV5vAHFyqkeuyFB5KVNGFOBuwjkUGKYc8wh9QfpVzAA=	237367
264280	FExKgjj6CsbToTubdZ+kGsOmUx3gCvZVJCdZPcdPNF4=	208934
808723	oo9tGpHvTredpg9JkHgYbZAuxcwtSpQxU5mA/zUbxY8=	182455

```
In [61]: all_words.sort_values('count', ascending=False).head(50000)['count'].sum()
```

```
Out[61]: 14961322
```

```
In [19]: content_train_features.isnull().sum()
```

```
Out[19]: x3     248263
         x4     248263
         x34     50846
         x35     50846
         x61        32
         x64     71061
         x65     71061
```

```
x91     32
x94     140619
x95     140619
dtype: int64
```

On the hashed words we have 979_749 unique words on 17_000_000 (1.7kk rows x 10 collumns) words giving 5.76% of uniques words on the total words. This show us that word can have a huge impact on the classifier because we have some words multiples times. But we have to take care of the NaN values and treat them.

Boolean variables

```
In [5]: bool_vars = meta[(meta.category == 'boolean')].index
        train_features[bool_vars].describe()
        train_features[bool_vars].isnull().sum()

Out[5]: x1      248190
        x2      248190
        x10     248263
        x11     248263
        x12     248263
        x13     248263
        x14     248263
        x24     248263
        x25     248263
        x26     248263
        x30     0
        x31     0
        x32     50772
        x33     50772
        x41     50846
        x42     50846
        x43     50846
        x44     50846
        x45     50846
        x55     50846
        x56     50846
        x57     50846
        x62     70978
        x63     70978
        x71     71061
        x72     71061
        x73     71061
        x74     71061
        x75     71061
        x85     71061
        x86     71061
        x87     71061
        x92     140526
```

```
x93     140526
x101    140619
x102    140619
x103    140619
x104    140619
x105    140619
x115    140619
x116    140619
x117    140619
x126    32
x127    32
x128    32
x129    32
x130    32
x140    32
x141    32
x142    32
dtype: int64
```

On the boolean values, only on 2 features we have no missing values. So we have to treat all this missing values here.

Labels variables

```
In [21]: total = train_labels.shape[0]
         for col in train_labels.columns:
             if col != 'id':
                 print(train_labels[col].value_counts(sort=True))
                 print('')
```

```
0    1689631
1    10369
Name: y1, dtype: int64

0    1698871
1    1129
Name: y2, dtype: int64

0    1664400
1    35600
Name: y3, dtype: int64

0    1677704
1    22296
Name: y4, dtype: int64

0    1699855
1    145
Name: y5, dtype: int64
```

```
0    1573102
1    126898
Name: y6, dtype: int64

0    1635569
1    64431
Name: y7, dtype: int64

0    1698519
1    1481
Name: y8, dtype: int64

0    1567117
1    132883
Name: y9, dtype: int64

0    1670709
1    29291
Name: y10, dtype: int64

0    1698432
1    1568
Name: y11, dtype: int64

0    1575878
1    124122
Name: y12, dtype: int64

0    1675185
1    24815
Name: y13, dtype: int64

0    1700000
Name: y14, dtype: int64

0    1695913
1    4087
Name: y15, dtype: int64

0    1681200
1    18800
Name: y16, dtype: int64

0    1699824
1    176
Name: y17, dtype: int64
```

```
0    1699704
1    296
Name: y18, dtype: int64

0    1698863
1    1137
Name: y19, dtype: int64

0    1695057
1    4943
Name: y20, dtype: int64

0    1687464
1    12536
Name: y21, dtype: int64

0    1689263
1    10737
Name: y22, dtype: int64

0    1698254
1    1746
Name: y23, dtype: int64

0    1671226
1    28774
Name: y24, dtype: int64

0    1695854
1    4146
Name: y25, dtype: int64

0    1681055
1    18945
Name: y26, dtype: int64

0    1683645
1    16355
Name: y27, dtype: int64

0    1683304
1    16696
Name: y28, dtype: int64

0    1646234
1    53766
Name: y29, dtype: int64
```

```
0    1659727
1    40273
Name: y30, dtype: int64

0    1649605
1    50395
Name: y31, dtype: int64

0    1606851
1    93149
Name: y32, dtype: int64

1    951246
0    748754
Name: y33, dtype: int64
```

We only have two types of response on labels 0 and 1, making a binary classification problem

```
In [22]: for col in train_labels.columns:
             if col != 'id':
                 total_1 = total - train_labels[col].value_counts(sort=True)[0]
                 perc = total_1 / total
                 print('Column {} has {} positive labels, {:.2%} of total'.format(col, total_1
```

```
Column y1 has 10369 positive labels, 0.61% of total
Column y2 has 1129 positive labels, 0.07% of total
Column y3 has 35600 positive labels, 2.09% of total
Column y4 has 22296 positive labels, 1.31% of total
Column y5 has 145 positive labels, 0.01% of total
Column y6 has 126898 positive labels, 7.46% of total
Column y7 has 64431 positive labels, 3.79% of total
Column y8 has 1481 positive labels, 0.09% of total
Column y9 has 132883 positive labels, 7.82% of total
Column y10 has 29291 positive labels, 1.72% of total
Column y11 has 1568 positive labels, 0.09% of total
Column y12 has 124122 positive labels, 7.30% of total
Column y13 has 24815 positive labels, 1.46% of total
Column y14 has 0 positive labels, 0.00% of total
Column y15 has 4087 positive labels, 0.24% of total
Column y16 has 18800 positive labels, 1.11% of total
Column y17 has 176 positive labels, 0.01% of total
Column y18 has 296 positive labels, 0.02% of total
Column y19 has 1137 positive labels, 0.07% of total
Column y20 has 4943 positive labels, 0.29% of total
Column y21 has 12536 positive labels, 0.74% of total
Column y22 has 10737 positive labels, 0.63% of total
```

```
Column y23 has 1746 positive labels, 0.10% of total
Column y24 has 28774 positive labels, 1.69% of total
Column y25 has 4146 positive labels, 0.24% of total
Column y26 has 18945 positive labels, 1.11% of total
Column y27 has 16355 positive labels, 0.96% of total
Column y28 has 16696 positive labels, 0.98% of total
Column y29 has 53766 positive labels, 3.16% of total
Column y30 has 40273 positive labels, 2.37% of total
Column y31 has 50395 positive labels, 2.96% of total
Column y32 has 93149 positive labels, 5.48% of total
Column y33 has 951246 positive labels, 55.96% of total
```

As we see most of the labels have few positive values and the last label has 55.96% of the total rows in positive values.

```
In [26]: train_labels.loc[:, train_labels.columns != 'id'].sum(axis=1).value_counts()
```

```
Out[26]: 1    1547755
         2    110870
         3    31909
         4    9321
         5    145
         dtype: int64
```

91.04% of text blocks have only one label and the rest has more than one label.

Data Quality Checks *Checking Missings Values*

```
In [23]: vars_with_missing = []
         for f in train_features.columns:
             missings = train_features[f].isnull().sum()
             if missings > 0:
                 vars_with_missing.append(f)
                 missings_perc = missings/train_features.shape[0]
                 category = meta.loc[f]['category']
                 dtype = meta.loc[f]['dtype']

                 print('Variable {} ({}, {}) has {} records ({:.2%}) with missing values'.forma

         print('In total, there are {} variables with missing values'.format(len(vars_with_miss
```

```
Variable x1 (boolean, object) has 248190 records (14.60%) with missing values
Variable x2 (boolean, object) has 248190 records (14.60%) with missing values
Variable x3 (content, object) has 248263 records (14.60%) with missing values
Variable x4 (content, object) has 248263 records (14.60%) with missing values
Variable x10 (boolean, object) has 248263 records (14.60%) with missing values
Variable x11 (boolean, object) has 248263 records (14.60%) with missing values
Variable x12 (boolean, object) has 248263 records (14.60%) with missing values
```

```
Variable x13 (boolean, object) has 248263 records (14.60%) with missing values
Variable x14 (boolean, object) has 248263 records (14.60%) with missing values
Variable x24 (boolean, object) has 248263 records (14.60%) with missing values
Variable x25 (boolean, object) has 248263 records (14.60%) with missing values
Variable x26 (boolean, object) has 248263 records (14.60%) with missing values
Variable x32 (boolean, object) has 50772 records (2.99%) with missing values
Variable x33 (boolean, object) has 50772 records (2.99%) with missing values
Variable x34 (content, object) has 50846 records (2.99%) with missing values
Variable x35 (content, object) has 50846 records (2.99%) with missing values
Variable x41 (boolean, object) has 50846 records (2.99%) with missing values
Variable x42 (boolean, object) has 50846 records (2.99%) with missing values
Variable x43 (boolean, object) has 50846 records (2.99%) with missing values
Variable x44 (boolean, object) has 50846 records (2.99%) with missing values
Variable x45 (boolean, object) has 50846 records (2.99%) with missing values
Variable x55 (boolean, object) has 50846 records (2.99%) with missing values
Variable x56 (boolean, object) has 50846 records (2.99%) with missing values
Variable x57 (boolean, object) has 50846 records (2.99%) with missing values
Variable x61 (content, object) has 32 records (0.00%) with missing values
Variable x62 (boolean, object) has 70978 records (4.18%) with missing values
Variable x63 (boolean, object) has 70978 records (4.18%) with missing values
Variable x64 (content, object) has 71061 records (4.18%) with missing values
Variable x65 (content, object) has 71061 records (4.18%) with missing values
Variable x71 (boolean, object) has 71061 records (4.18%) with missing values
Variable x72 (boolean, object) has 71061 records (4.18%) with missing values
Variable x73 (boolean, object) has 71061 records (4.18%) with missing values
Variable x74 (boolean, object) has 71061 records (4.18%) with missing values
Variable x75 (boolean, object) has 71061 records (4.18%) with missing values
Variable x85 (boolean, object) has 71061 records (4.18%) with missing values
Variable x86 (boolean, object) has 71061 records (4.18%) with missing values
Variable x87 (boolean, object) has 71061 records (4.18%) with missing values
Variable x91 (content, object) has 32 records (0.00%) with missing values
Variable x92 (boolean, object) has 140526 records (8.27%) with missing values
Variable x93 (boolean, object) has 140526 records (8.27%) with missing values
Variable x94 (content, object) has 140619 records (8.27%) with missing values
Variable x95 (content, object) has 140619 records (8.27%) with missing values
Variable x101 (boolean, object) has 140619 records (8.27%) with missing values
Variable x102 (boolean, object) has 140619 records (8.27%) with missing values
Variable x103 (boolean, object) has 140619 records (8.27%) with missing values
Variable x104 (boolean, object) has 140619 records (8.27%) with missing values
Variable x105 (boolean, object) has 140619 records (8.27%) with missing values
Variable x115 (boolean, object) has 140619 records (8.27%) with missing values
Variable x116 (boolean, object) has 140619 records (8.27%) with missing values
Variable x117 (boolean, object) has 140619 records (8.27%) with missing values
Variable x126 (boolean, object) has 32 records (0.00%) with missing values
Variable x127 (boolean, object) has 32 records (0.00%) with missing values
Variable x128 (boolean, object) has 32 records (0.00%) with missing values
Variable x129 (boolean, object) has 32 records (0.00%) with missing values
Variable x130 (boolean, object) has 32 records (0.00%) with missing values
```

```
Variable x140 (boolean, object) has 32 records (0.00%) with missing values
Variable x141 (boolean, object) has 32 records (0.00%) with missing values
Variable x142 (boolean, object) has 32 records (0.00%) with missing values
In total, there are 58 variables with missing values
```

Some missing values variables repeat the quatity of missing values, this can be because of the relational text blocks, so if this text block is the leftmost block on the document, they will not have some block at the left. And this can go for other directions too.

Checking the cardinality of the int variables

Cardinality means the differents values of a variable, so we will see which feature will became dummy variables.

```
In [24]: for f in int_train_features:
             dist_values = int_train_features[f].value_counts().shape[0]
             print('Variable {} has {} distinct values'.format(f, dist_values))
```

```
Variable x15 has 97 distinct values
Variable x17 has 119 distinct values
Variable x18 has 108 distinct values
Variable x22 has 498 distinct values
Variable x23 has 419 distinct values
Variable x27 has 193 distinct values
Variable x46 has 122 distinct values
Variable x48 has 167 distinct values
Variable x49 has 110 distinct values
Variable x53 has 499 distinct values
Variable x54 has 419 distinct values
Variable x58 has 149 distinct values
Variable x76 has 127 distinct values
Variable x78 has 184 distinct values
Variable x79 has 109 distinct values
Variable x83 has 498 distinct values
Variable x84 has 417 distinct values
Variable x88 has 141 distinct values
Variable x106 has 109 distinct values
Variable x108 has 123 distinct values
Variable x109 has 109 distinct values
Variable x113 has 500 distinct values
Variable x114 has 419 distinct values
Variable x118 has 159 distinct values
Variable x131 has 125 distinct values
Variable x133 has 158 distinct values
Variable x134 has 110 distinct values
Variable x138 has 500 distinct values
Variable x139 has 420 distinct values
Variable x143 has 493 distinct values
```

At this point i can't see if I will treat this variables as categorical and transform in dummy variables or treat them as quatitative variables.

1.3.2 Exploratory Visualization

1. Section **??**
2. Section **??**
3. Section **??**

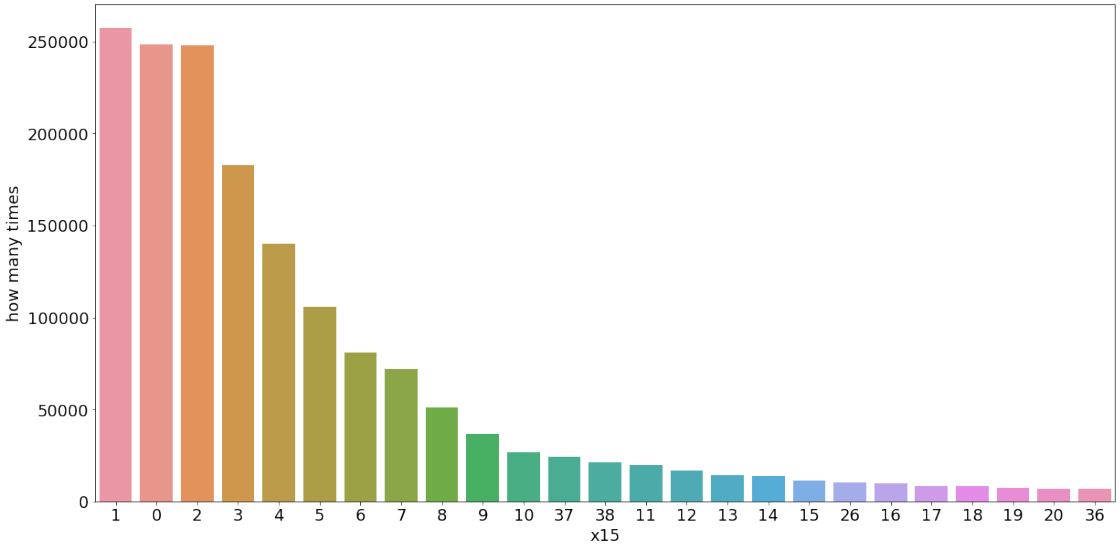
In this section, we will explore visually the dataset trying to summarizes and extracts relevants characteristics about the data.

Numerical int variables

```
In [27]: for v in int_features:
             feature = train_features[v]
             print("Variable {} has {} different values".format(v, feature.value_counts().shape
             print("Ploting only the 25 largest values")
             pl.categorical(train_features, v)
             print(feature.describe().apply(lambda x: format(x, 'f')))
             print('\n\n\n')
```

```
Variable x15 has 97 different values
Ploting only the 25 largest values
```

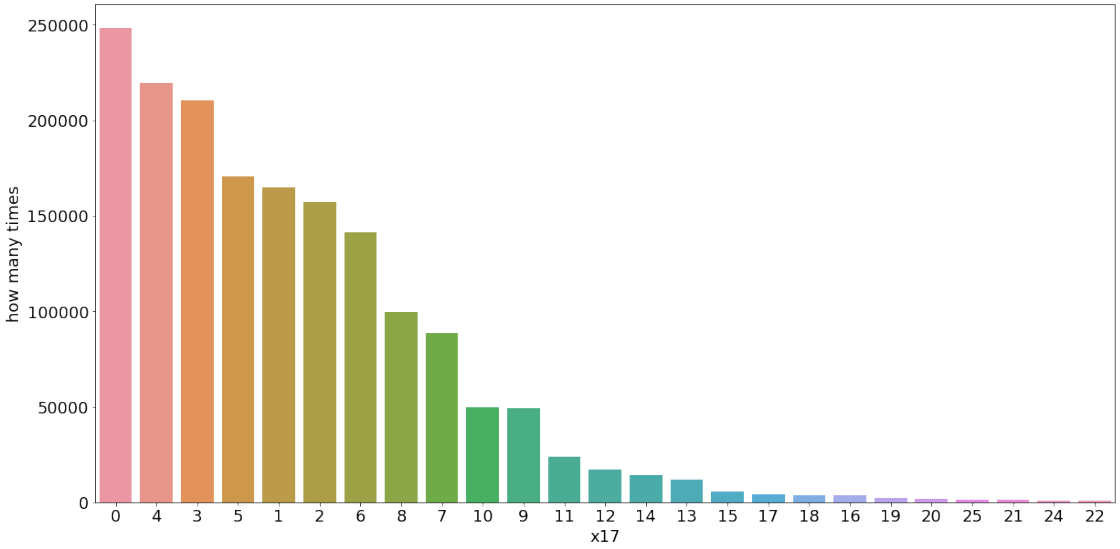
```
<Figure size 432x288 with 0 Axes>
```



```
count     1700000.000000
mean            6.154404
std             8.957511
min             0.000000
25%             1.000000
50%             3.000000
75%             7.000000
max           153.000000
Name: x15, dtype: object
```

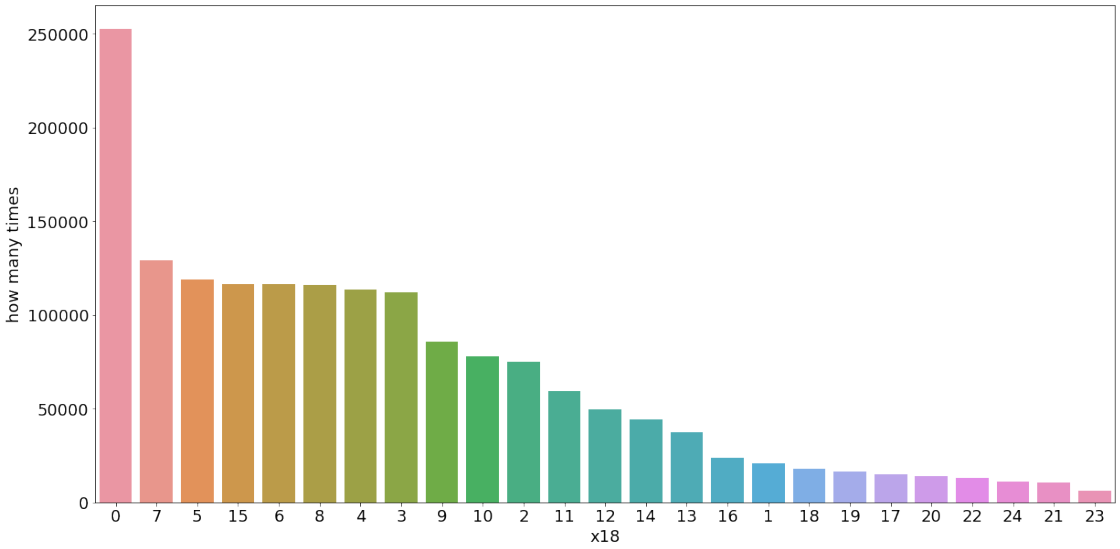
```
Variable x17 has 119 different values
Ploting only the 25 largest values
```

```
<Figure size 432x288 with 0 Axes>
```

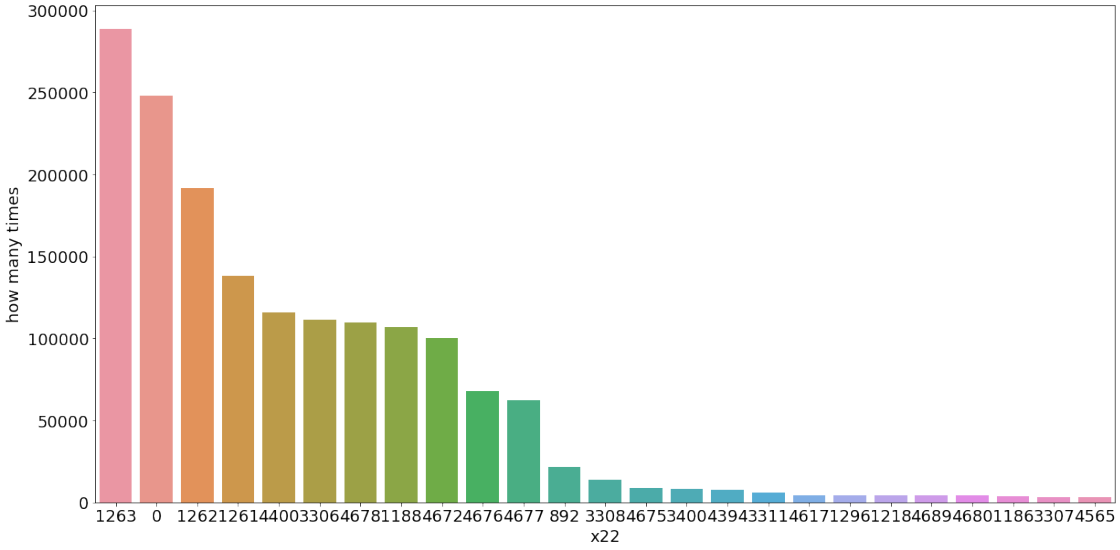



```
count     1700000.000000
mean            4.487084
std             4.623426
min             0.000000
25%             2.000000
50%             4.000000
75%             6.000000
max           237.000000
Name: x17, dtype: object
```

```
Variable x18 has 108 different values
Ploting only the 25 largest values

<Figure size 432x288 with 0 Axes>
```



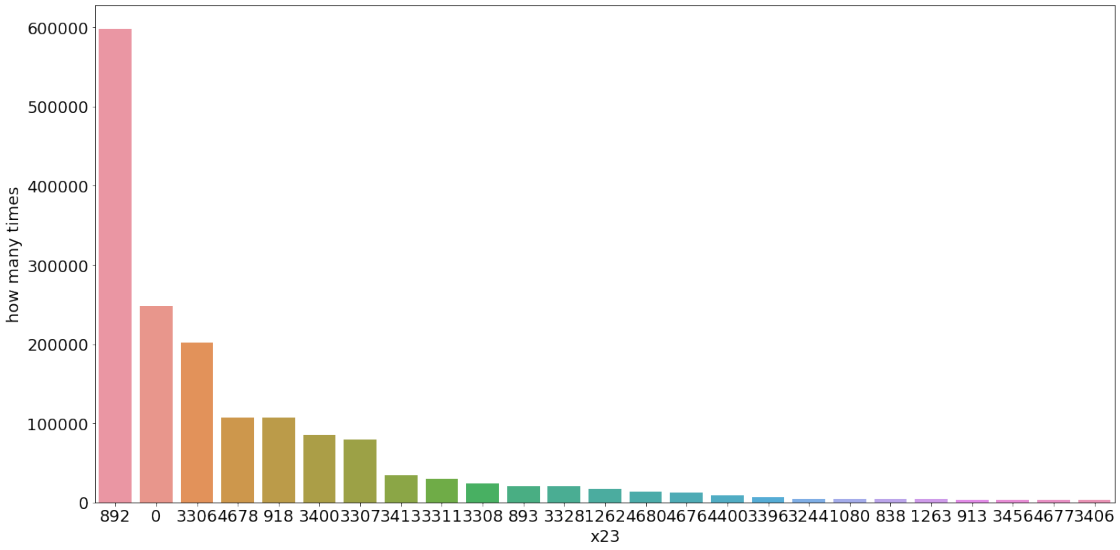

```
count    1700000.000000
mean           8.096322
std            7.123864
min            0.000000
25%            3.000000
50%            7.000000
75%           11.000000
max          219.000000
Name: x18, dtype: object


Variable x22 has 498 different values
Ploting only the 25 largest values

<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean      2301.595118
std       1745.119509
min       0.000000
25%       1261.000000
50%       1263.000000
75%       4400.000000
max       19500.000000
Name: x22, dtype: object
```

```
Variable x23 has 419 different values
Ploting only the 25 largest values
```

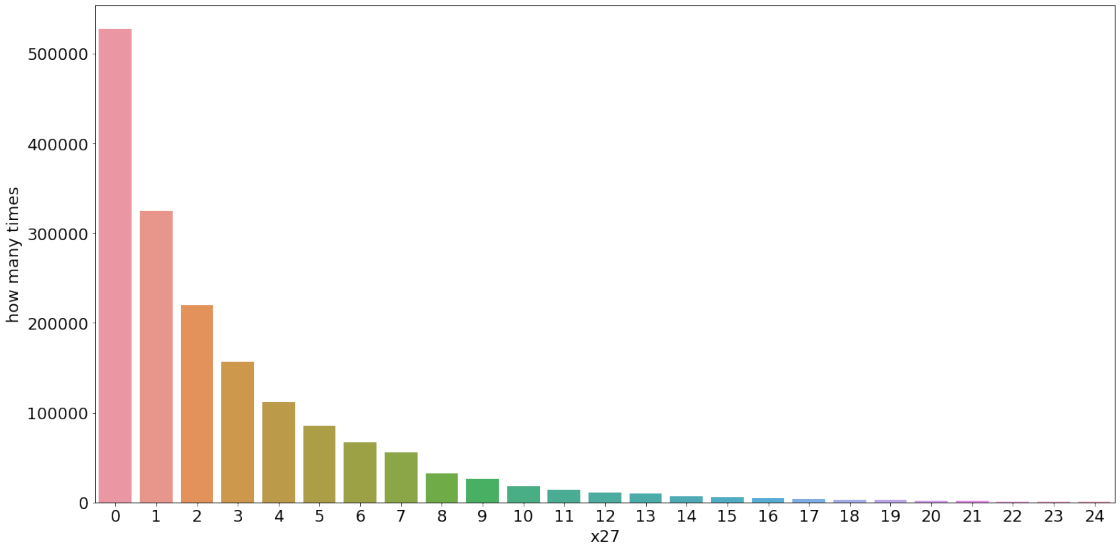
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean      1874.765323
std       1517.990813
min       0.000000
25%       892.000000
50%       892.000000
75%       3307.000000
max       14167.000000
Name: x23, dtype: object
```

```
Variable x27 has 193 different values
Ploting only the 25 largest values
```

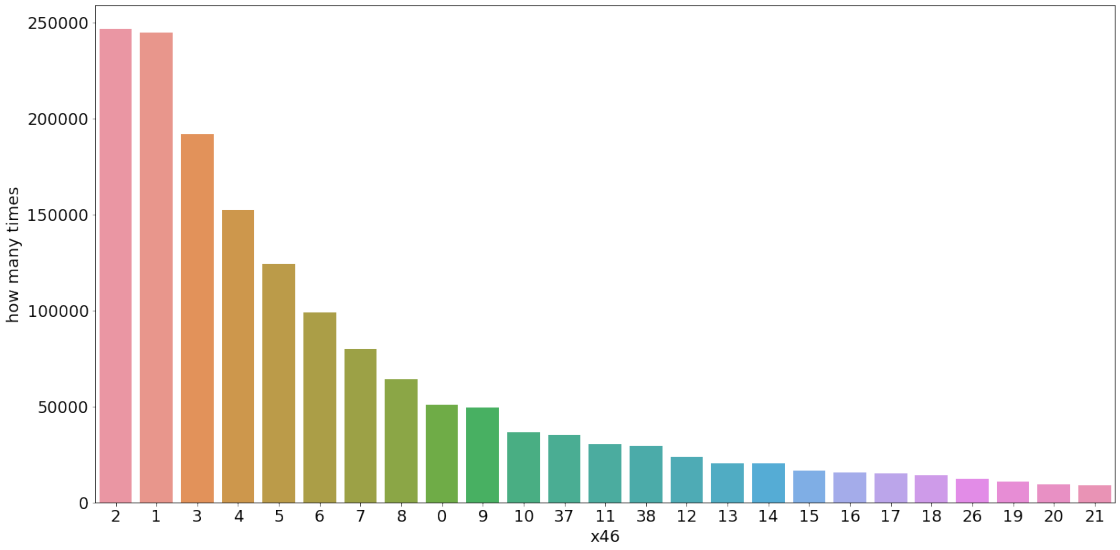
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean            2.814097
std             4.409801
min            -1.000000
25%             0.000000
50%             1.000000
75%             4.000000
max           672.000000
Name: x27, dtype: object
```

```
Variable x46 has 122 different values
Ploting only the 25 largest values
```

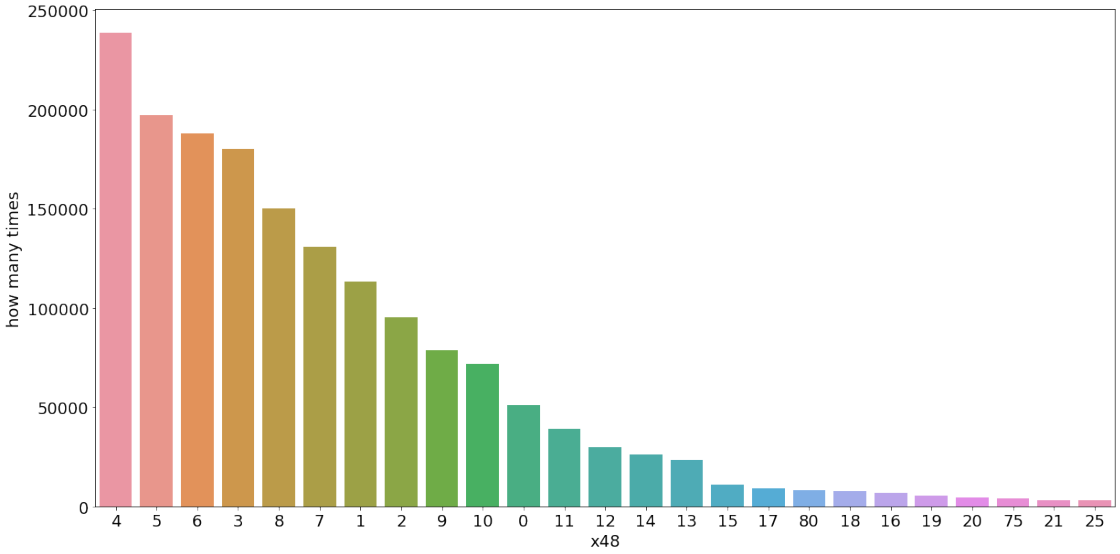
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean            8.151020
std            10.360501
min             0.000000
25%             2.000000
50%             4.000000
75%             9.000000
max           153.000000
Name: x46, dtype: object
```

```
Variable x48 has 167 different values
Ploting only the 25 largest values
```

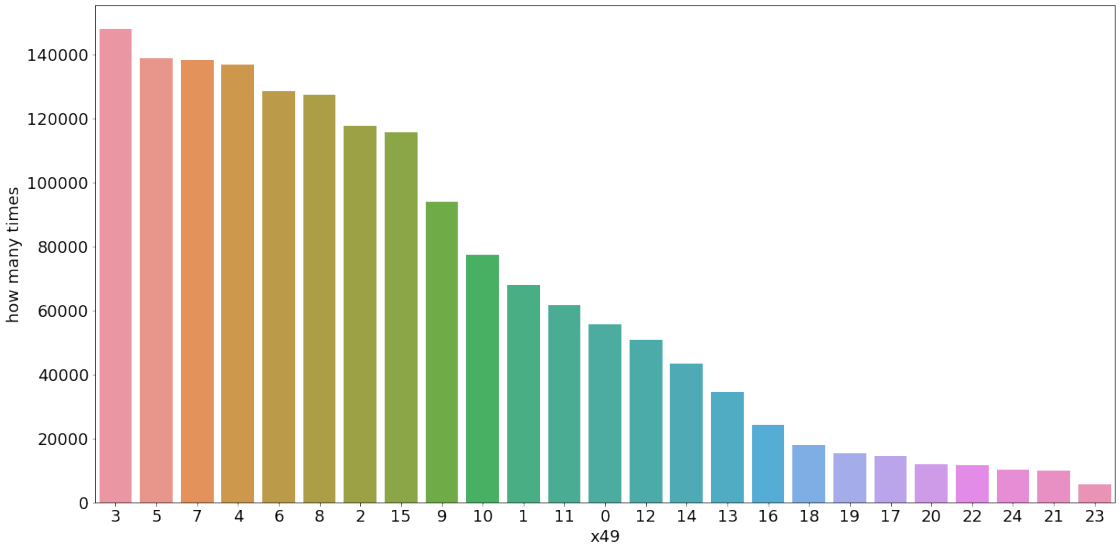
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean            6.973005
std             9.311837
min             0.000000
25%             3.000000
50%             5.000000
75%             8.000000
max           371.000000
Name: x48, dtype: object
```

```
Variable x49 has 110 different values
Ploting only the 25 largest values
```

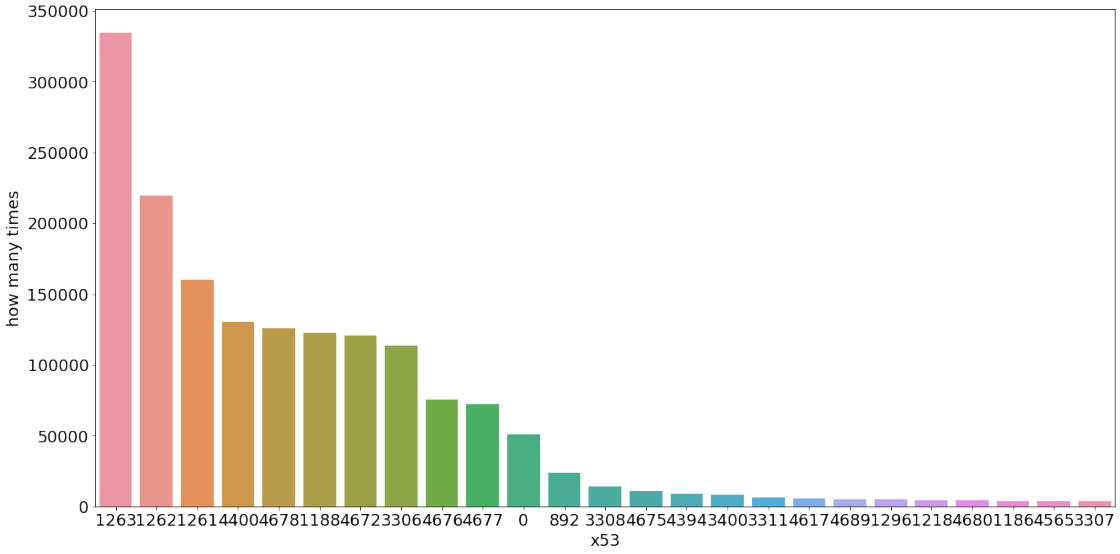
```
<Figure size 432x288 with 0 Axes>
```




```
count    1700000.000000
mean           8.363310
std            6.656968
min            0.000000
25%            4.000000
50%            7.000000
75%           11.000000
max          219.000000
Name: x49, dtype: object
```

```
Variable x53 has 499 different values
Ploting only the 25 largest values
```

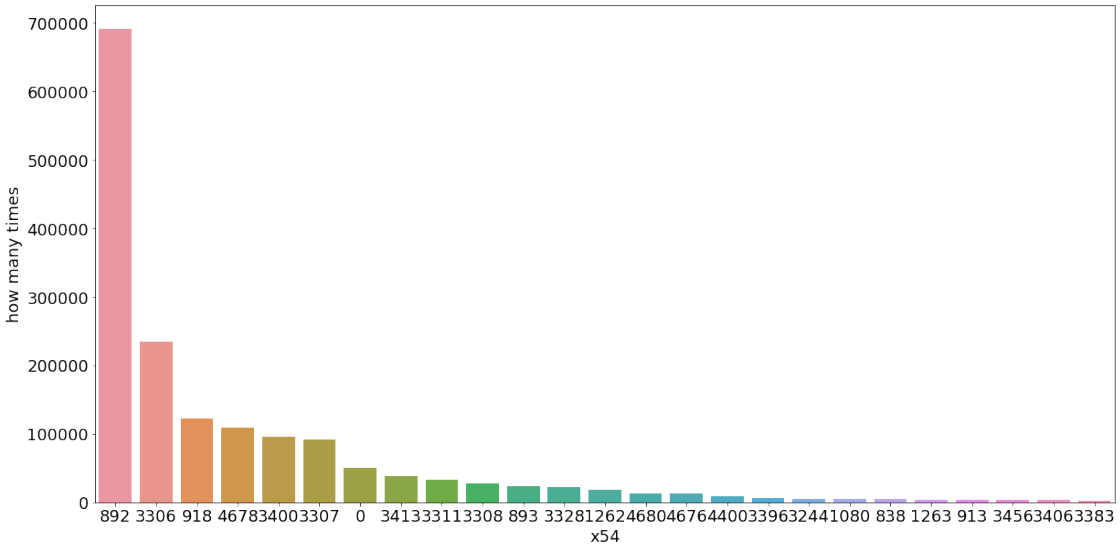
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean      2605.914796
std       1632.476755
min       0.000000
25%       1262.000000
50%       1263.000000
75%       4659.000000
max       19500.000000
Name: x53, dtype: object
```

```
Variable x54 has 419 different values
Ploting only the 25 largest values
```

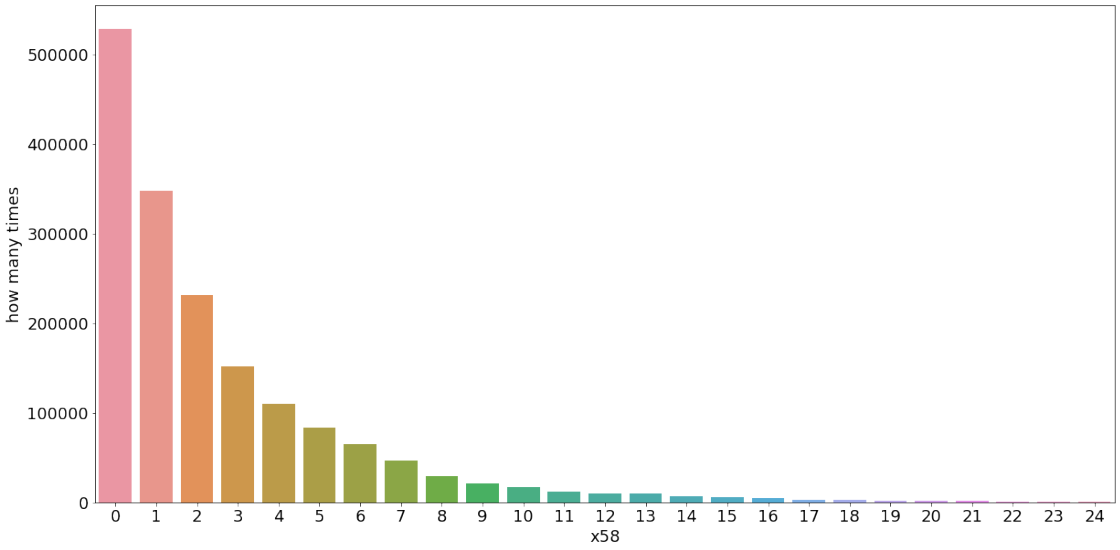
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean      2098.132814
std       1421.664275
min       0.000000
25%       892.000000
50%       918.000000
75%       3307.000000
max       14167.000000
Name: x54, dtype: object
```

```
Variable x58 has 149 different values
Ploting only the 25 largest values
```

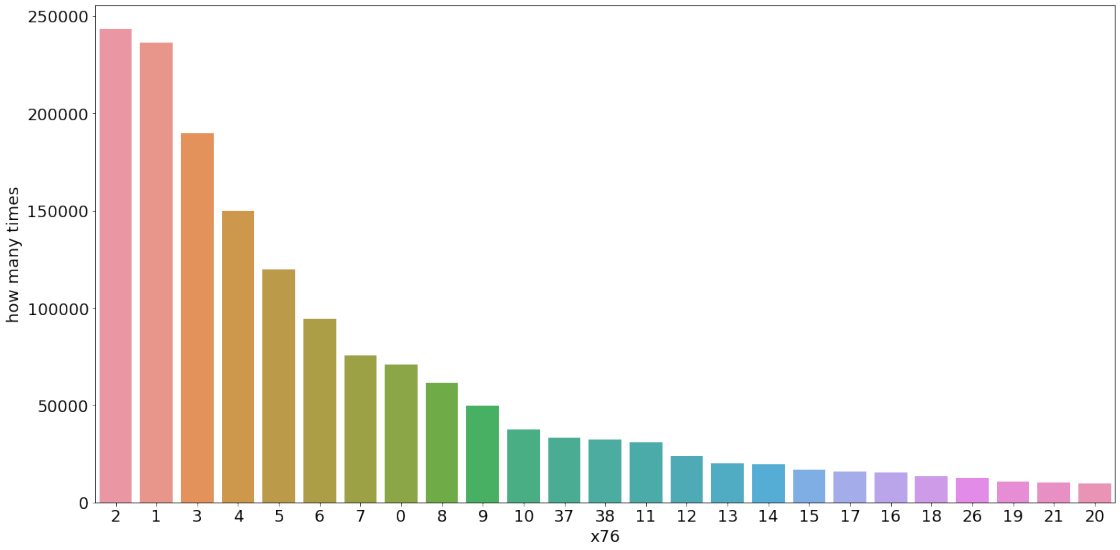
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean            2.640614
std             3.823310
min            -1.000000
25%             0.000000
50%             1.000000
75%             4.000000
max           337.000000
Name: x58, dtype: object
```

```
Variable x76 has 127 different values
Ploting only the 25 largest values
```

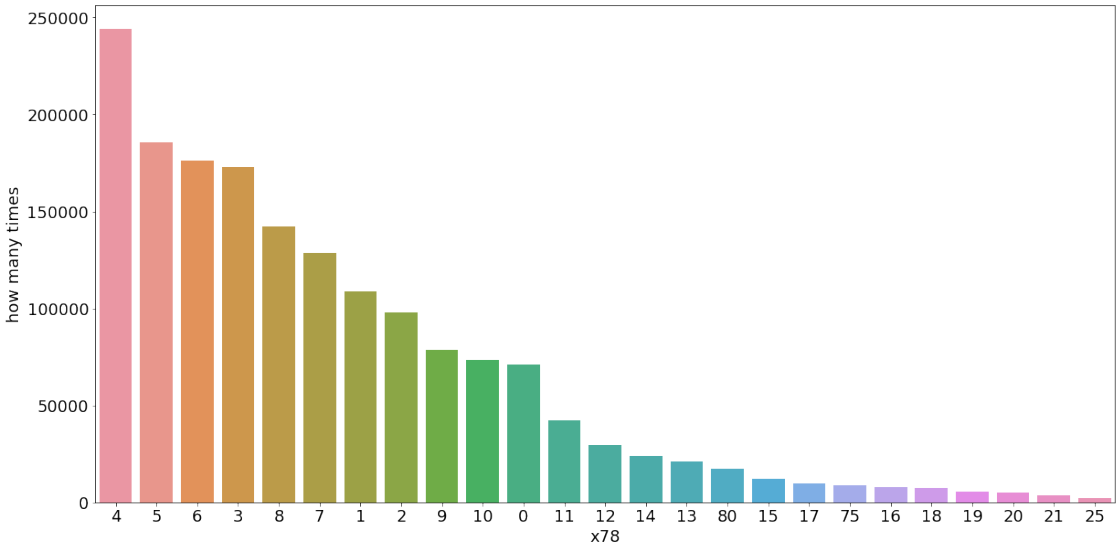
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean            8.259205
std            10.557784
min             0.000000
25%             2.000000
50%             4.000000
75%             9.000000
max           153.000000
Name: x76, dtype: object
```

```
Variable x78 has 184 different values
Ploting only the 25 largest values
```

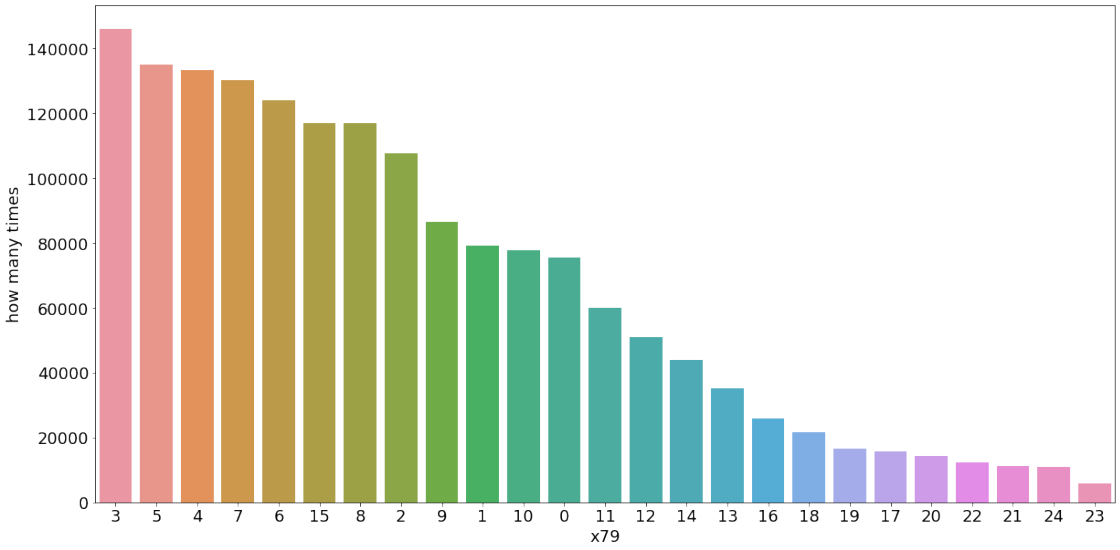
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean            7.502141
std            11.295486
min             0.000000
25%             3.000000
50%             5.000000
75%             8.000000
max           271.000000
Name: x78, dtype: object
```

```
Variable x79 has 109 different values
Ploting only the 25 largest values
```

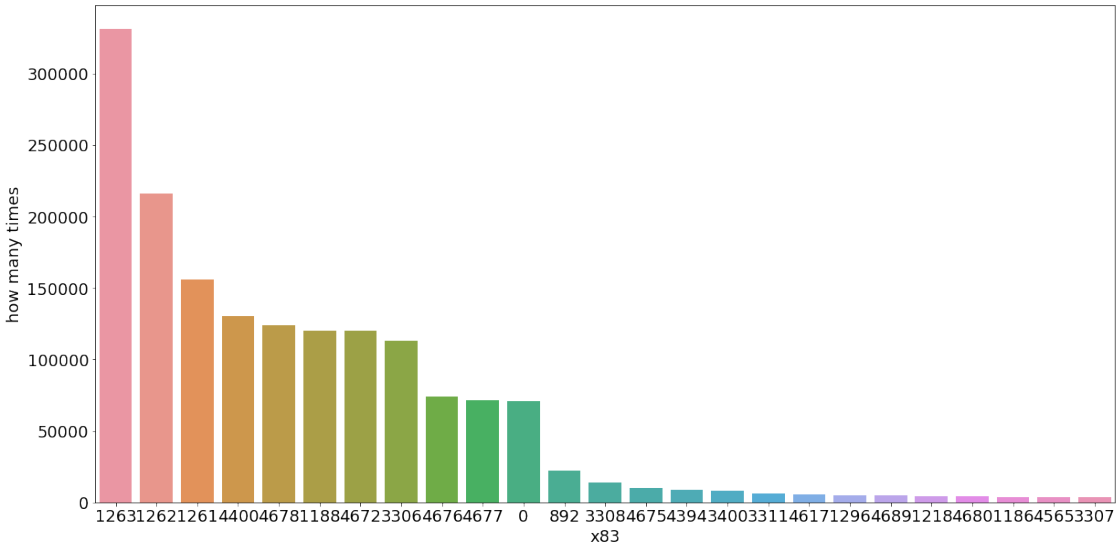
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean            8.401614
std             6.834330
min             0.000000
25%             4.000000
50%             7.000000
75%            12.000000
max           219.000000
Name: x79, dtype: object
```

```
Variable x83 has 498 different values
Ploting only the 25 largest values
```

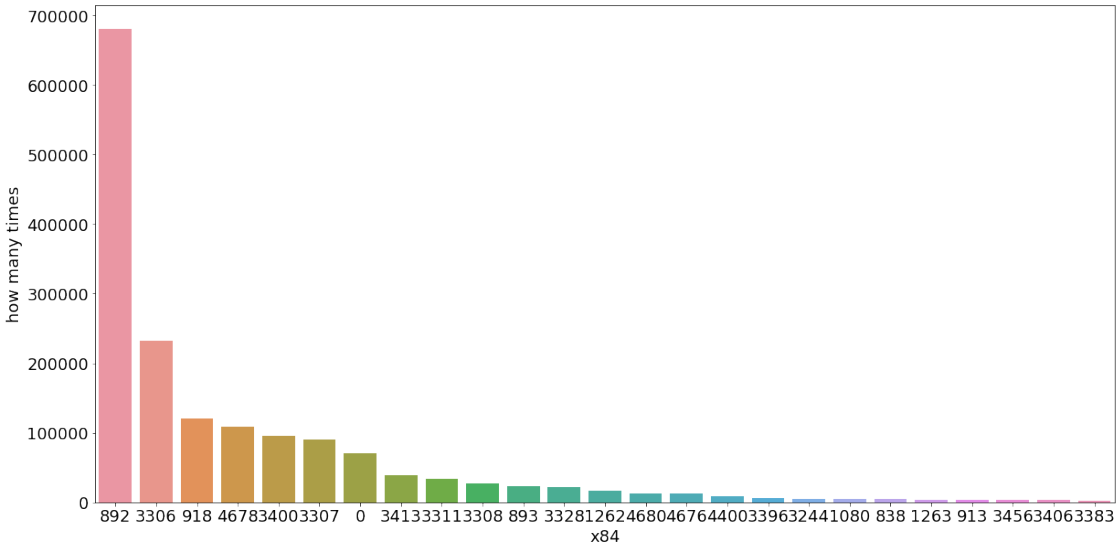
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean      2581.685180
std       1648.193261
min       0.000000
25%       1262.000000
50%       1263.000000
75%       4643.000000
max       19500.000000
Name: x83, dtype: object
```

```
Variable x84 has 417 different values
Ploting only the 25 largest values
```

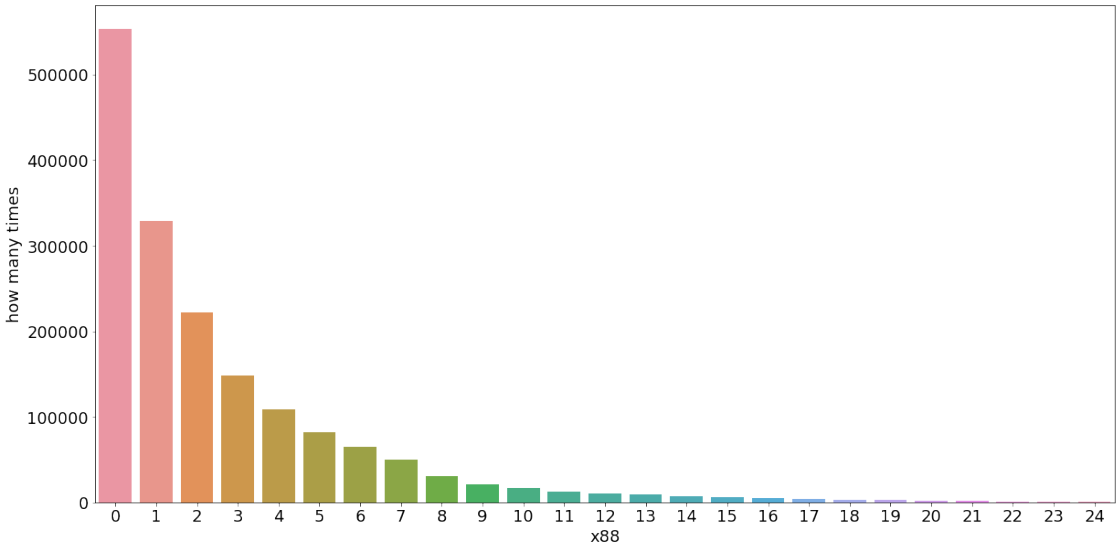
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean      2079.390638
std       1433.160352
min       0.000000
25%       892.000000
50%       918.000000
75%       3307.000000
max       14167.000000
Name: x84, dtype: object
```

```
Variable x88 has 141 different values
Ploting only the 25 largest values
```

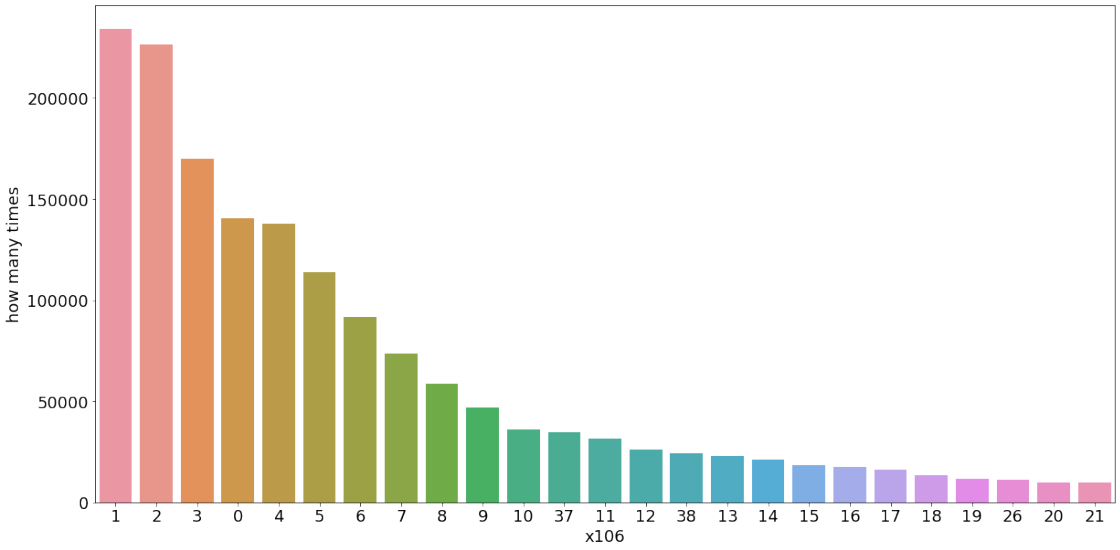
```
<Figure size 432x288 with 0 Axes>
```




```
count    1700000.000000
mean           2.685396
std            3.921535
min           -1.000000
25%            0.000000
50%            1.000000
75%            4.000000
max          284.000000
Name: x88, dtype: object
```

```
Variable x106 has 109 different values
Ploting only the 25 largest values
```

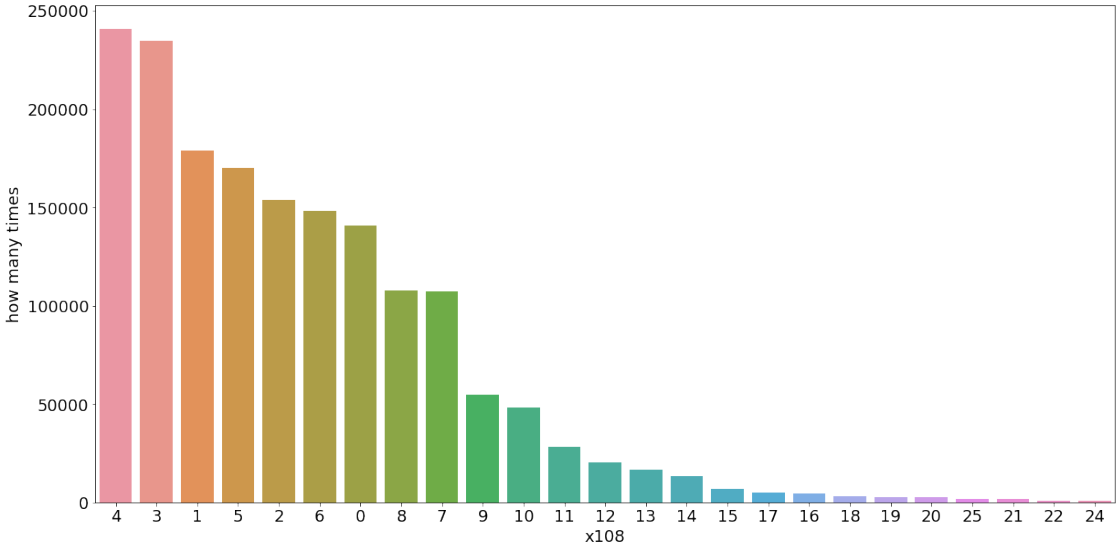
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean            7.851039
std            10.019375
min             0.000000
25%             2.000000
50%             4.000000
75%             9.000000
max           153.000000
Name: x106, dtype: object
```

```
Variable x108 has 123 different values
Ploting only the 25 largest values
```

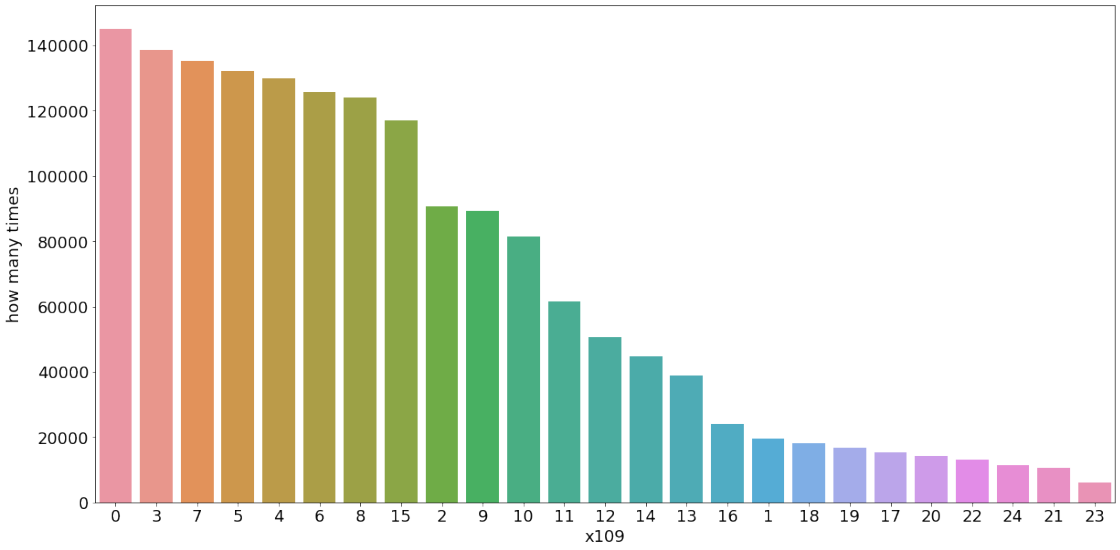
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean            4.854556
std             4.483374
min             0.000000
25%             2.000000
50%             4.000000
75%             7.000000
max           243.000000
Name: x108, dtype: object
```

```
Variable x109 has 109 different values
Ploting only the 25 largest values
```

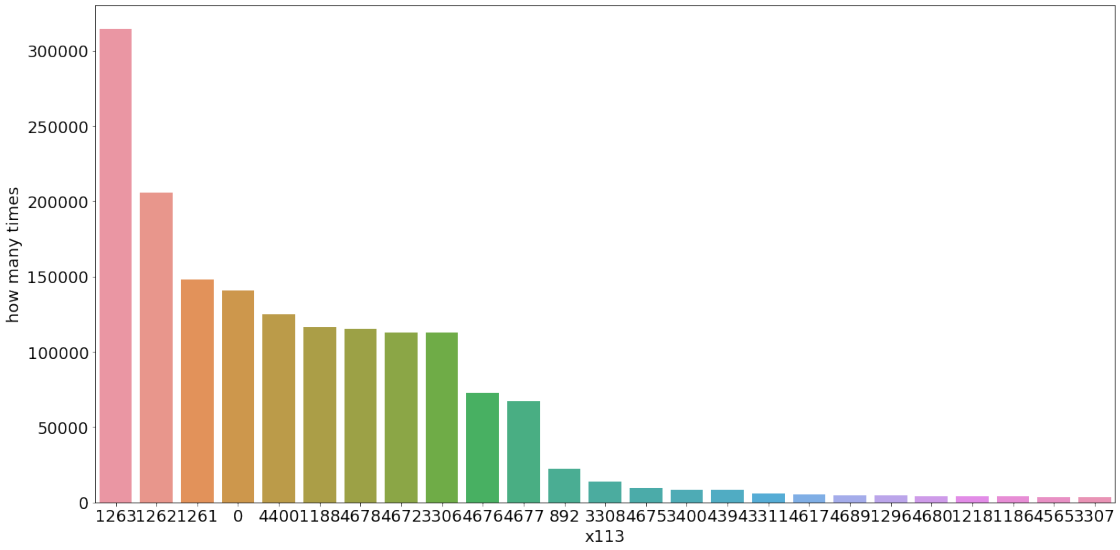
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean            8.419647
std             6.886586
min             0.000000
25%             4.000000
50%             7.000000
75%            12.000000
max           219.000000
Name: x109, dtype: object
```

```
Variable x113 has 500 different values
Ploting only the 25 largest values
```

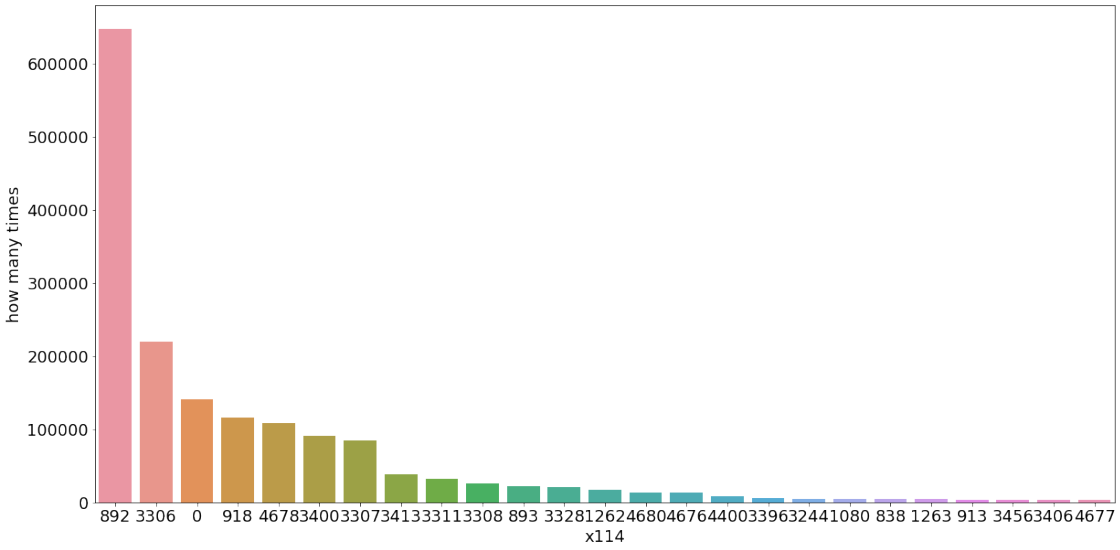
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean      2469.780722
std       1691.322508
min       0.000000
25%       1261.000000
50%       1263.000000
75%       4400.000000
max       19500.000000
Name: x113, dtype: object
```

```
Variable x114 has 419 different values
Ploting only the 25 largest values
```

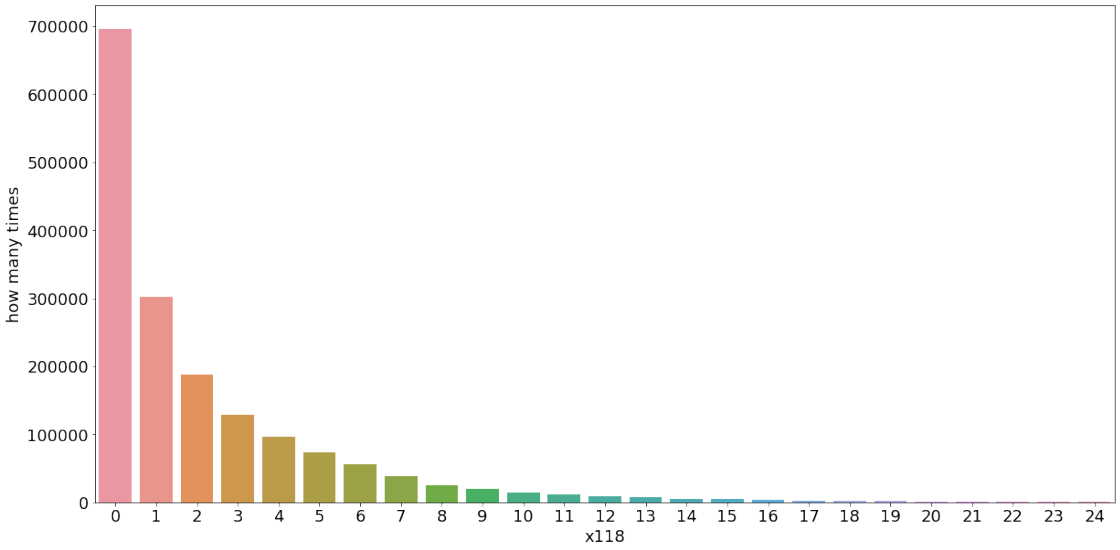
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean      1998.761406
std       1470.972957
min       0.000000
25%       892.000000
50%       918.000000
75%       3307.000000
max       14167.000000
Name: x114, dtype: object
```

```
Variable x118 has 159 different values
Ploting only the 25 largest values
```

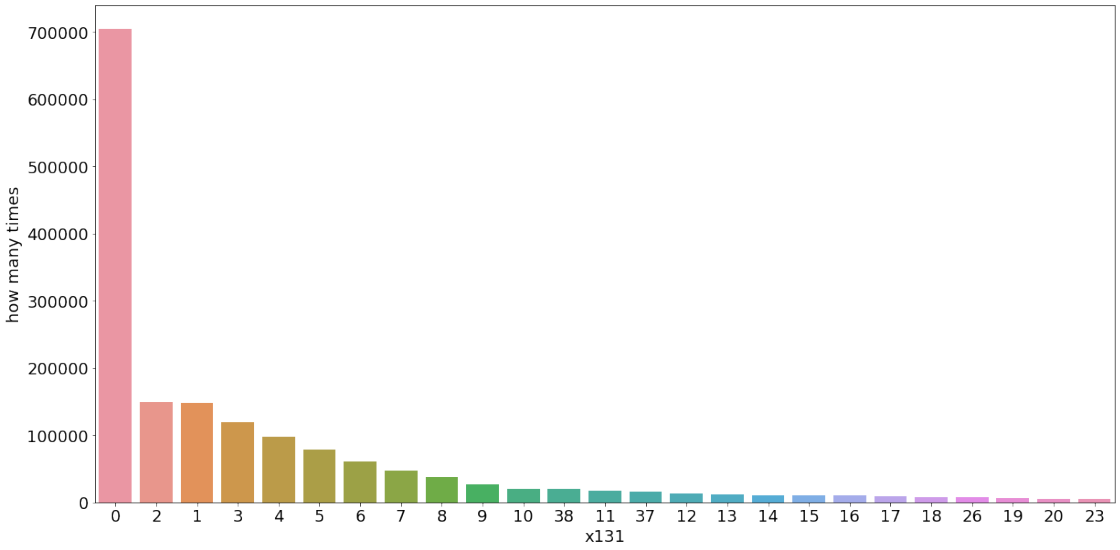
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean            2.314082
std             3.914509
min            -1.000000
25%             0.000000
50%             1.000000
75%             3.000000
max           410.000000
Name: x118, dtype: object
```

```
Variable x131 has 125 different values
Ploting only the 25 largest values
```

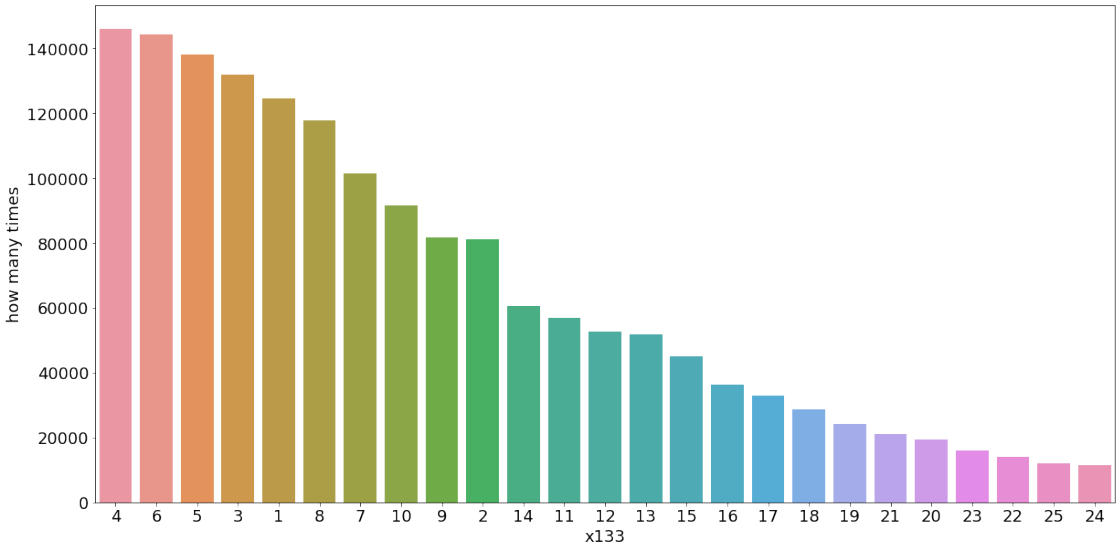
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean            4.809295
std             8.966942
min             0.000000
25%             0.000000
50%             1.000000
75%             5.000000
max           153.000000
Name: x131, dtype: object
```

```
Variable x133 has 158 different values
Ploting only the 25 largest values
```

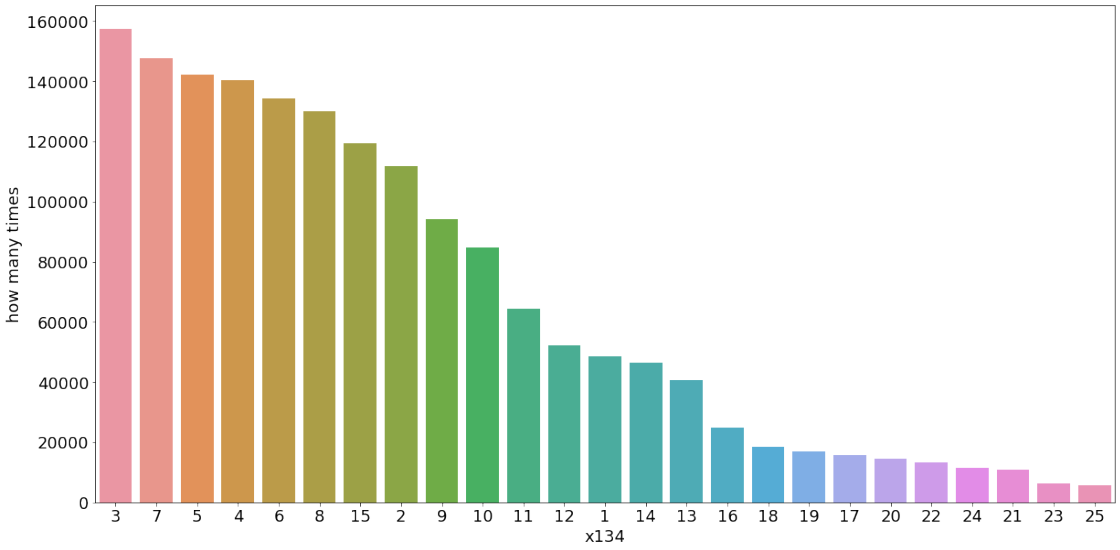
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean            9.301809
std             7.725215
min             0.000000
25%             4.000000
50%             7.000000
75%            13.000000
max           301.000000
Name: x133, dtype: object
```

```
Variable x134 has 110 different values
Ploting only the 25 largest values
```

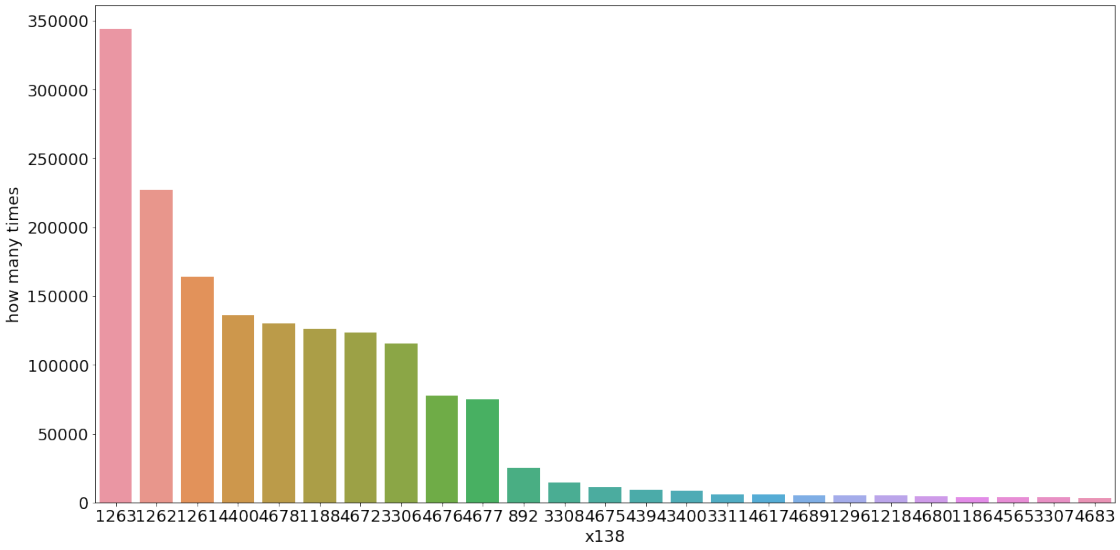
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean            8.842868
std             6.665332
min             0.000000
25%             4.000000
50%             7.000000
75%            12.000000
max           219.000000
Name: x134, dtype: object
```

```
Variable x138 has 500 different values
Ploting only the 25 largest values
```

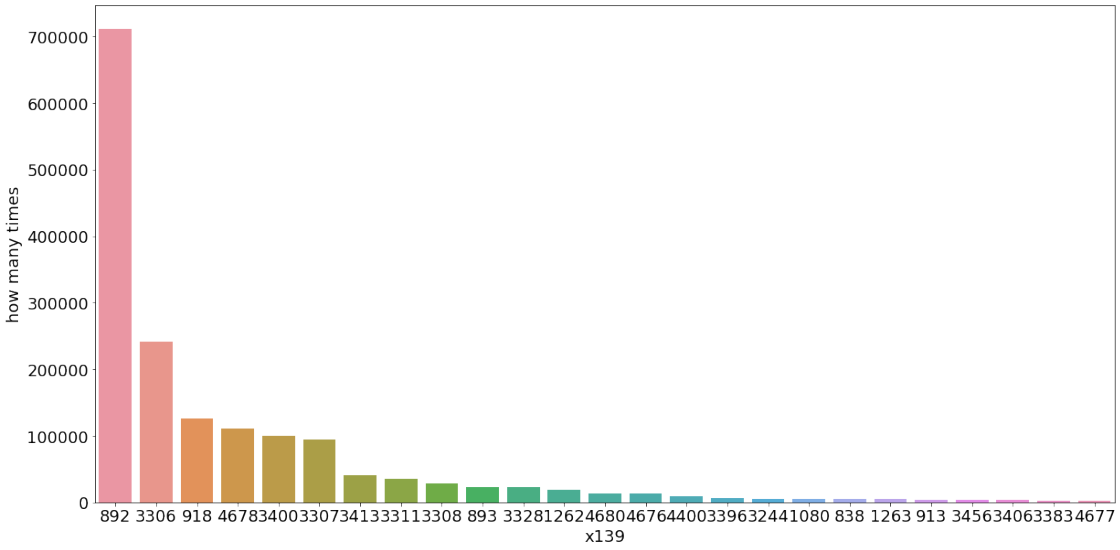
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean         2688.457461
std          1591.809367
min             0.000000
25%          1262.000000
50%          1263.000000
75%          4672.000000
max         19500.000000
Name: x138, dtype: object
```

```
Variable x139 has 420 different values
Ploting only the 25 largest values
```

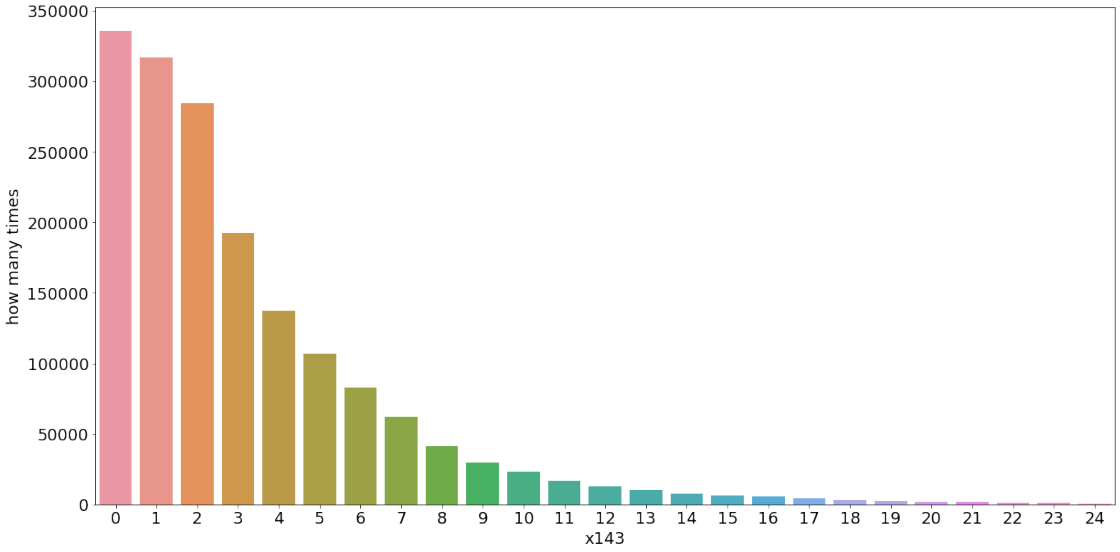
```
<Figure size 432x288 with 0 Axes>
```




```
count     1700000.000000
mean         2163.422956
std          1392.787044
min             0.000000
25%           892.000000
50%           918.000000
75%          3308.000000
max         14167.000000
Name: x139, dtype: object
```

```
Variable x143 has 493 different values
Ploting only the 25 largest values
```

```
<Figure size 432x288 with 0 Axes>
```



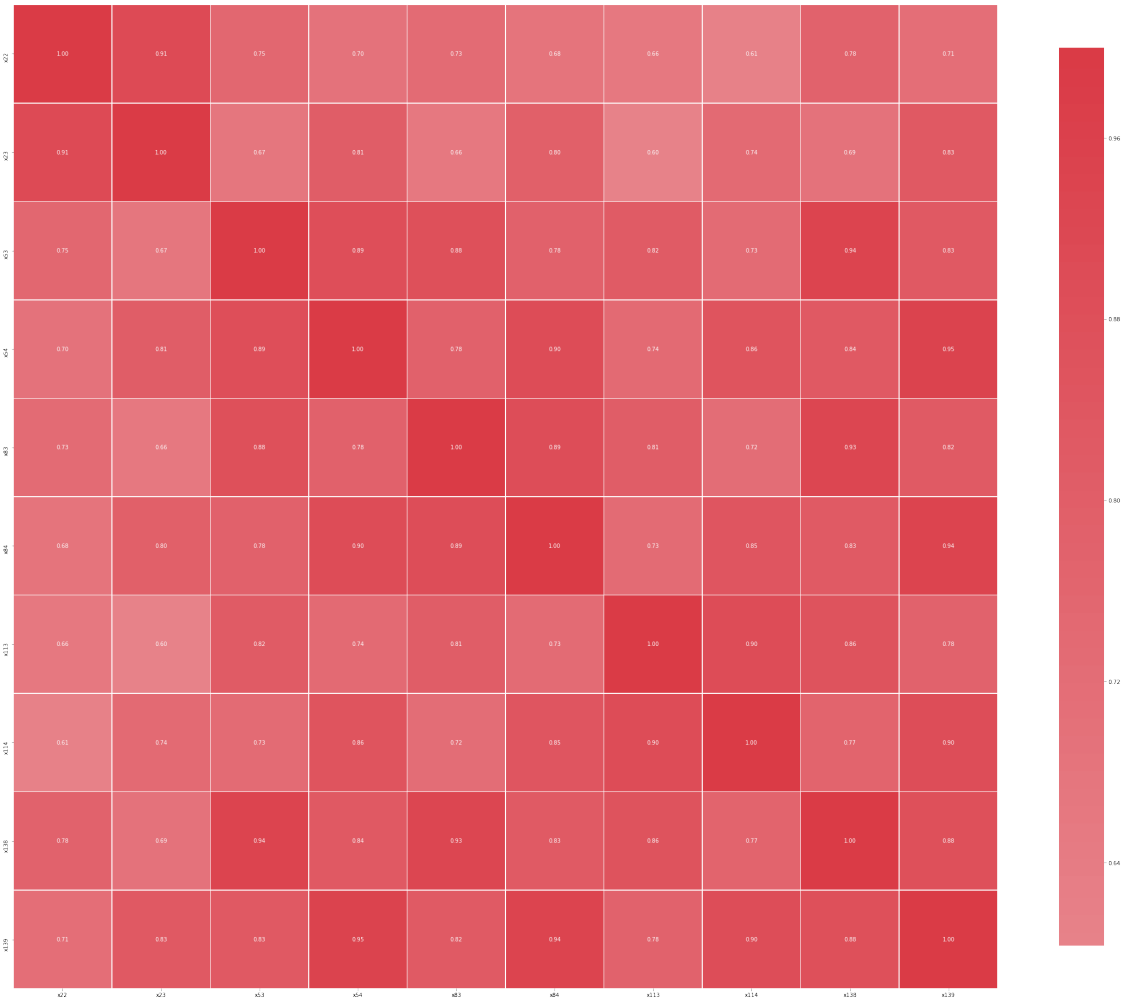

```
count     1700000.000000
mean            3.632134
std             9.424120
min            -1.000000
25%             1.000000
50%             2.000000
75%             5.000000
max          1219.000000
Name: x143, dtype: object
```

The numerical int variables show three different kinds of patterns: a 25% on 892, a 50% on 1263 and the rest that feel likes quantitative variables. I will classify the first as the height category, the second as the width category and the third as quantitative category.

```
In [4]: meta.at[['x23', 'x54', 'x84', 'x114', 'x139'], 'category'] = 'height'
        meta.at[['x22', 'x53', 'x83', 'x113', 'x138'], 'category'] = 'width'
        meta.at[['x15', 'x17', 'x18', 'x27', 'x46', 'x48', 'x49', 'x58', 'x76', 'x78', 'x79',
        meta[(meta.category == 'width')]

Out[4]:           role category  dtype
        varname
        x22      input  width    int64
        x53      input  width    int64
        x83      input  width    int64
```

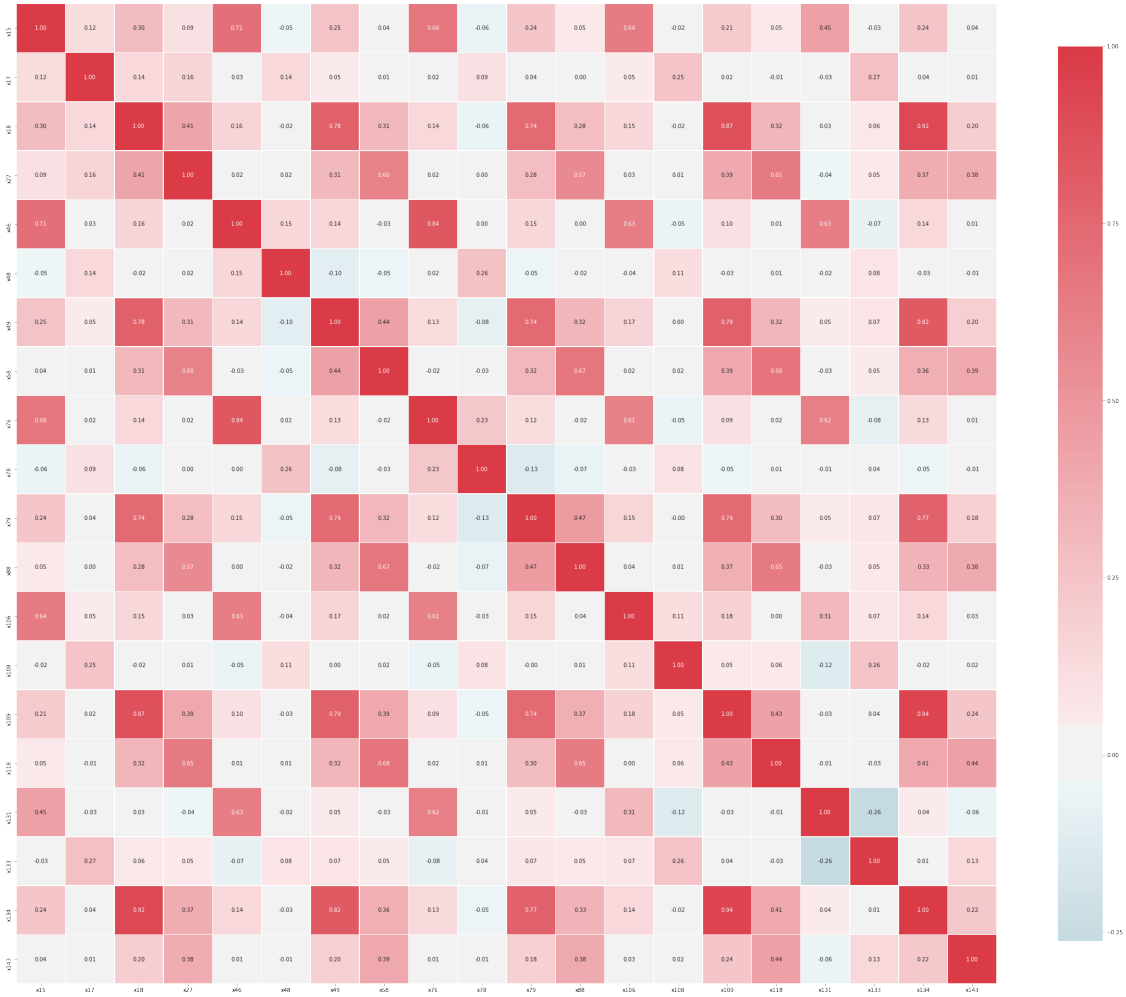
```
x113     input  width    int64
x138     input  width    int64
```

```
In [34]: pl.correlation_map(train_features, meta[(meta.category == 'height') | (meta.category =
```



As I suspect, the correlation between the width and height category is very linear.

```
In [35]: pl.correlation_map(train_features, meta[(meta.category == 'quantitative')].index)
```



```
In [16]: pca_features = train_features.sample(frac=0.5)
         pca = PCA(n_components=5)
         pca.fit(pca_features[meta[(meta.category == 'height') | (meta.category == 'width')].in
         [ "{:0.5f}".format(x) for x in pca.explained_variance_ratio_ ]

Out[16]: ['0.81348', '0.07019', '0.05062', '0.03623', '0.02164']
```

As component principal analysis with 5 components can explain 99,21% of the variance in the height/width part of the dataset, this can be used in the implementation part of the algorithms.

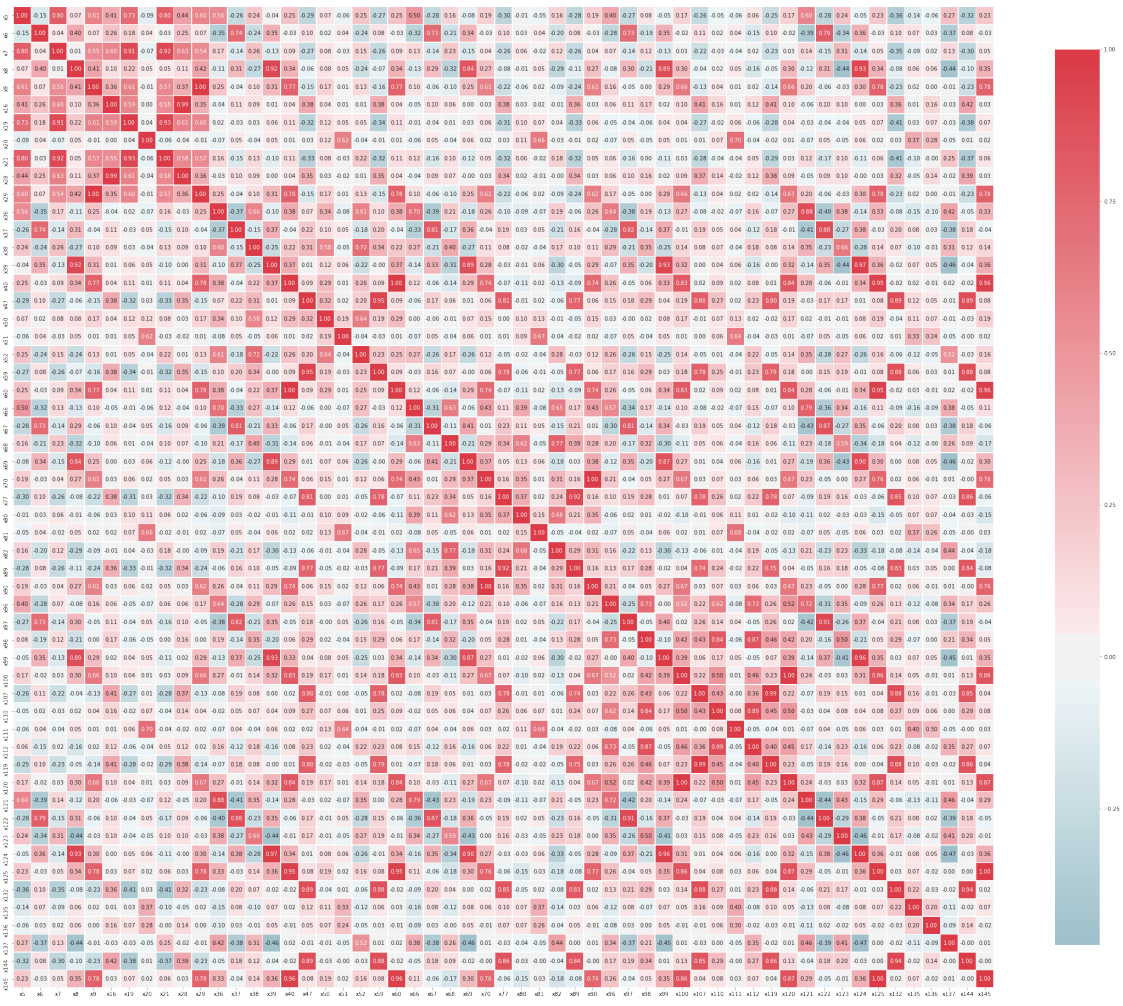
```
In [23]: pca_features = train_features.sample(frac=0.5)
         pca = PCA(n_components=10)
         pca.fit(pca_features[meta[(meta.category == 'quantitative')].index])
         [ "{:0.5f}".format(x) for x in pca.explained_variance_ratio_ ]

Out[23]: ['0.30023',
          '0.18144',
```

```
'0.12165',
'0.07665',
'0.06814',
'0.06394',
'0.03789',
'0.02542',
'0.02390',
'0.02135']
```

As component principal analysis with 10 components can explain 92,06% of the variance in the quantitative part of the dataset, this can be used in the implementation part of the algorithms.

Numerical float variables Checking the correlations between interval variables. A heatmap is a good way to visualize the correlation between variables.

```
In [32]: pl.correlation_map(train_features, float_features)
```



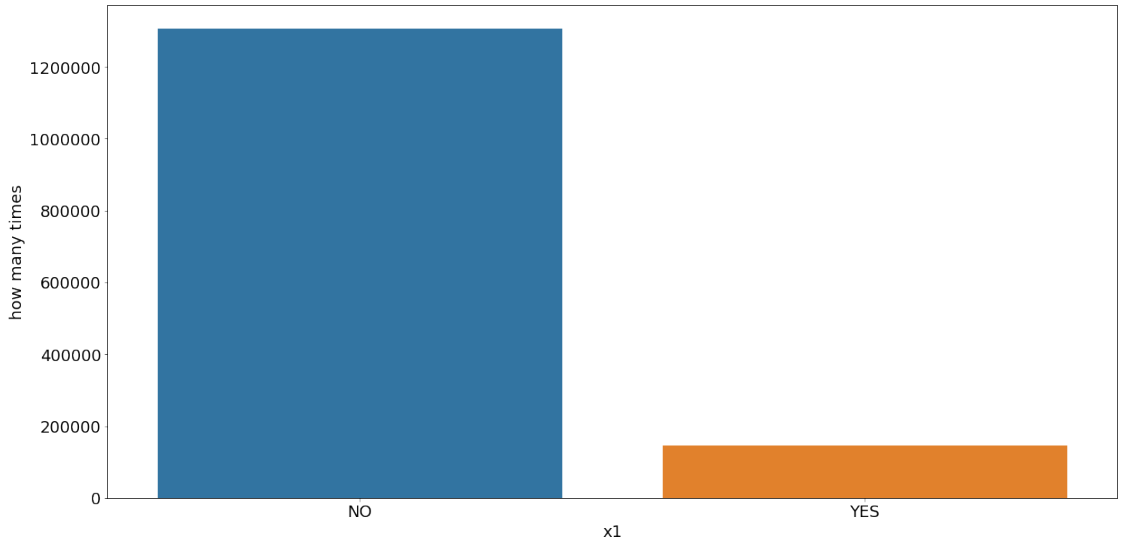
As we can see, that has many features with a nice linear correlation, which can make this part of the dataset a good call for a PCA to see how much the component analysis can explain the linear correlation.

```
In [12]: pca_features = train_features.sample(frac=0.5)
         pca = PCA(n_components=10)
         pca.fit(pca_features[(float_features)])
         [ "{:0.5f}".format(x) for x in pca.explained_variance_ratio_ ]

Out[12]: ['0.32163',
          '0.13221',
          '0.10925',
          '0.09541',
          '0.09278',
          '0.05833',
          '0.03506',
          '0.03238',
          '0.02477',
          '0.01815']
```

As component principal analysis with 10 components can explain 91.99% of the variance in the float part of the dataset, this can be used in the implementation part of the algorithms.

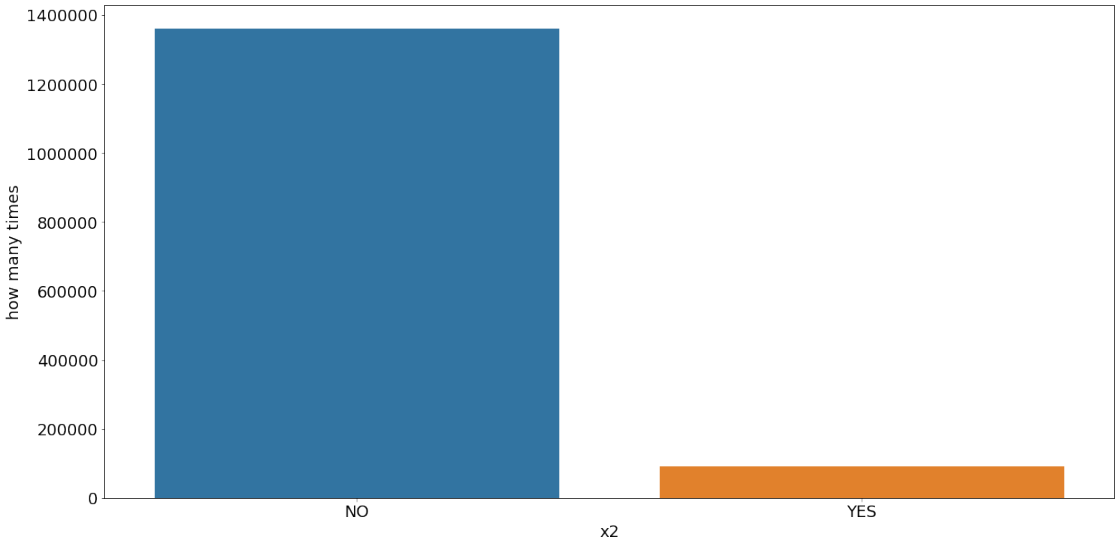
Boolean variables

```
In [13]: for v in bool_vars:
             feature = train_features[v]
             pl.categorical(train_features, v)
             print(feature.describe())

<Figure size 432x288 with 0 Axes>
```



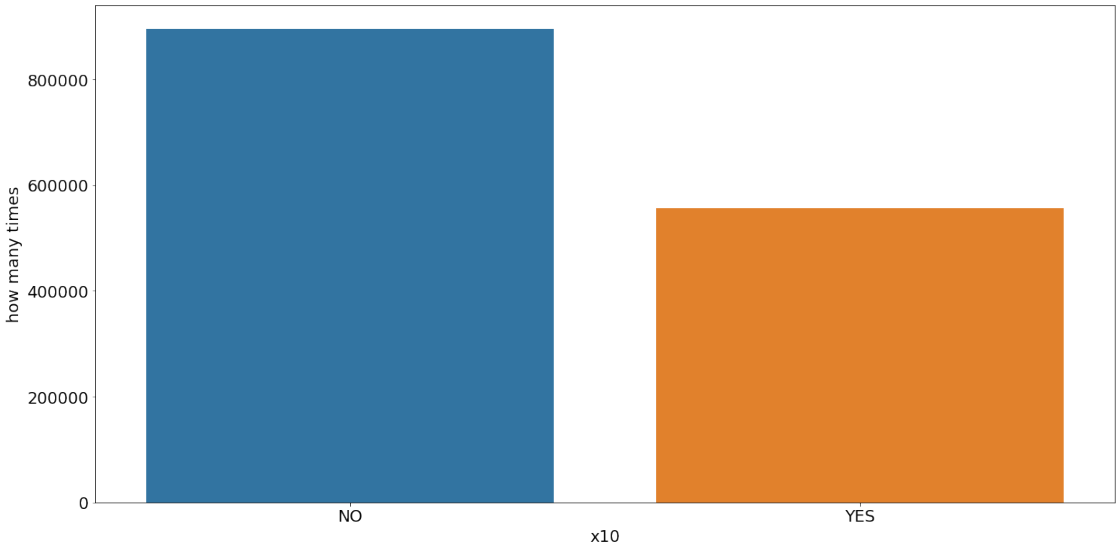
```
count      1451810
unique     2
top        NO
freq       1306048
Name: x1, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



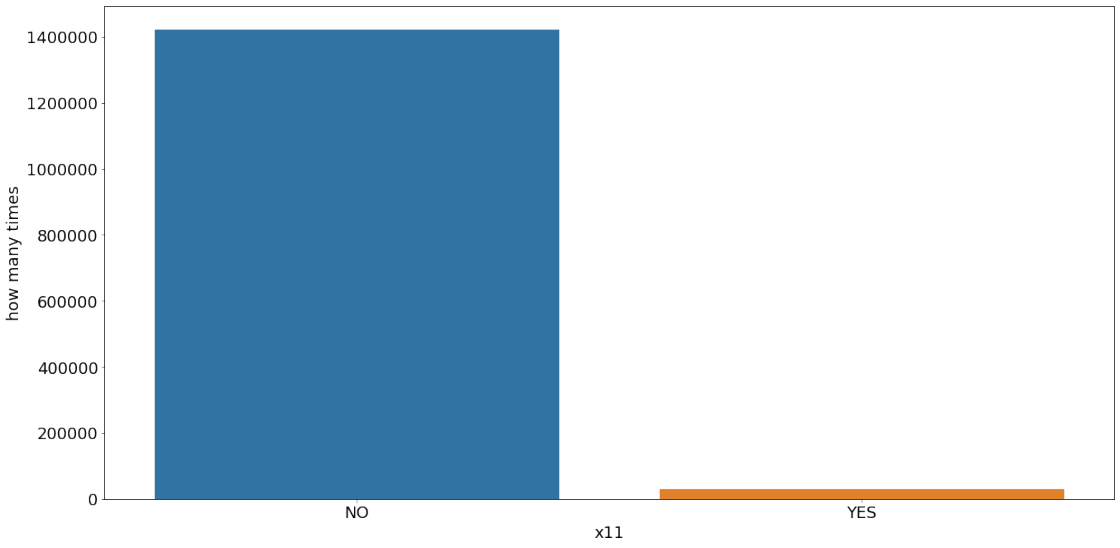

```
count      1451810
unique     2
top        NO
freq       1360034
Name: x2, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



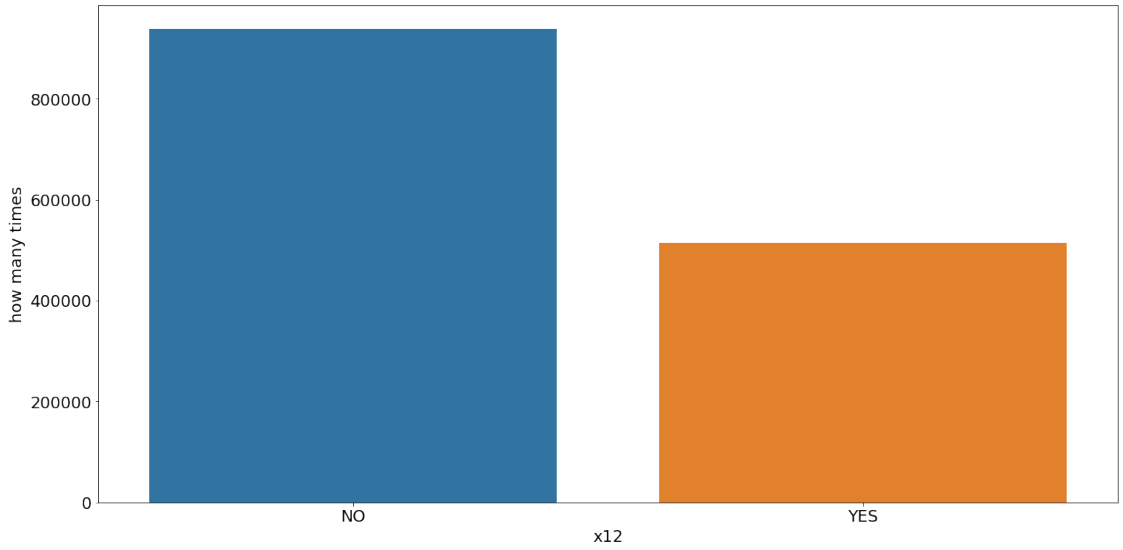

```
count      1451737
unique     2
top        NO
freq       895085
Name: x10, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



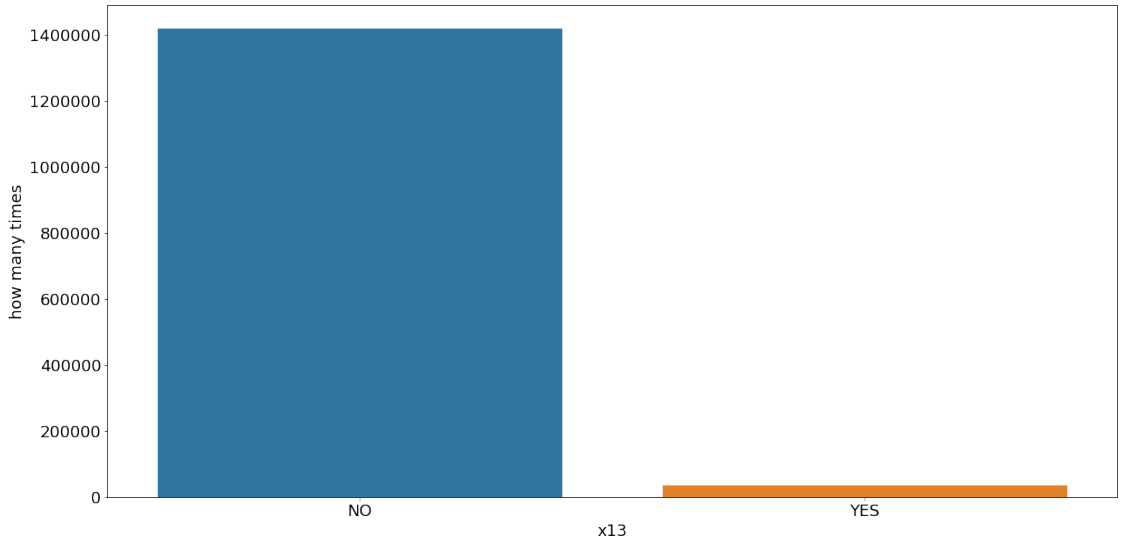
```
count     1451737
unique    2
top       NO
freq      1421672
Name: x11, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



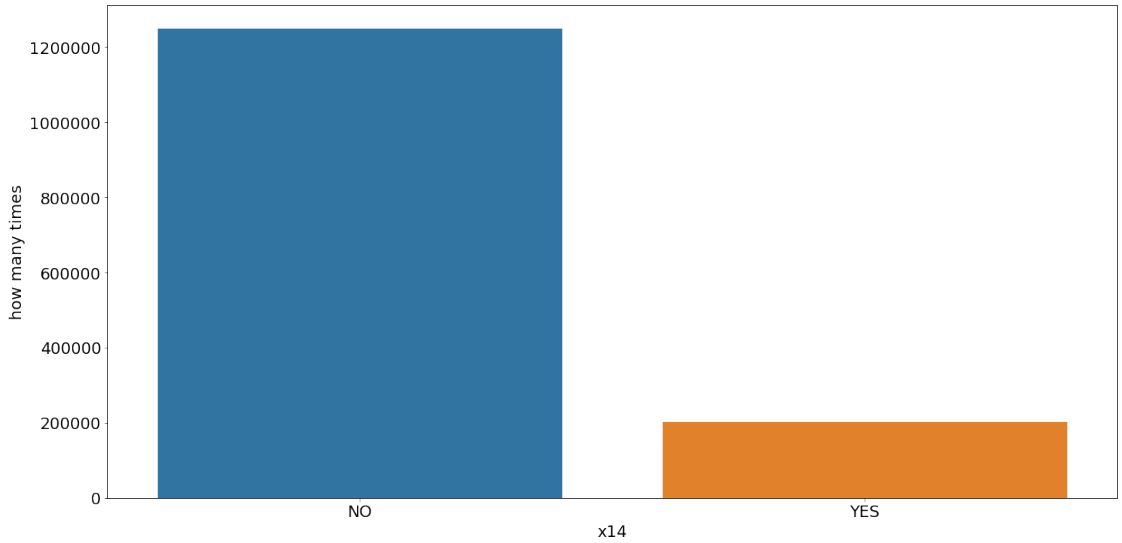

```
count     1451737
unique    2
top       NO
freq      937844
Name: x12, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



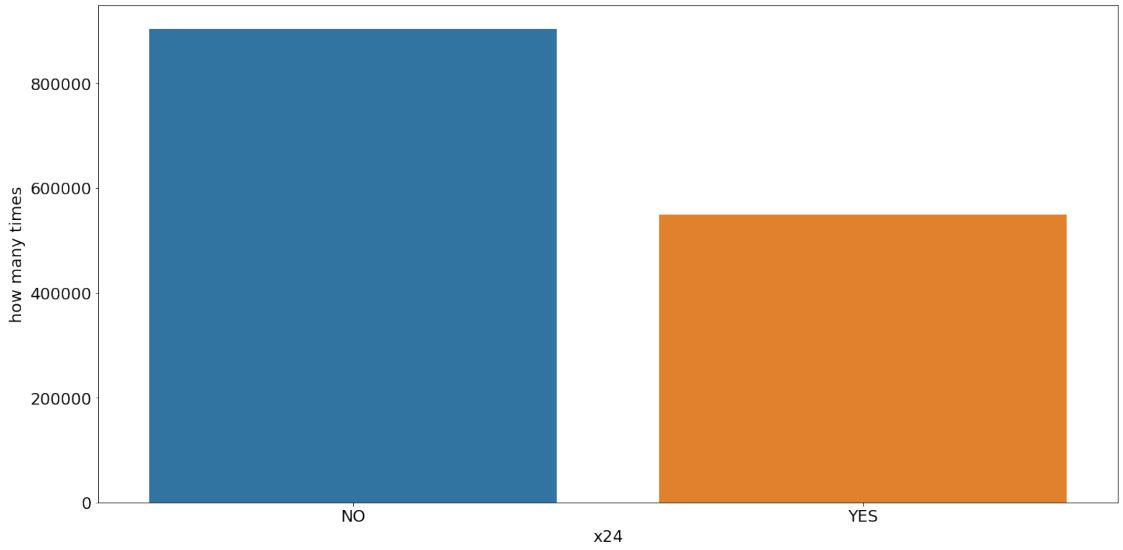

```
count      1451737
unique     2
top        NO
freq       1417787
Name: x13, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



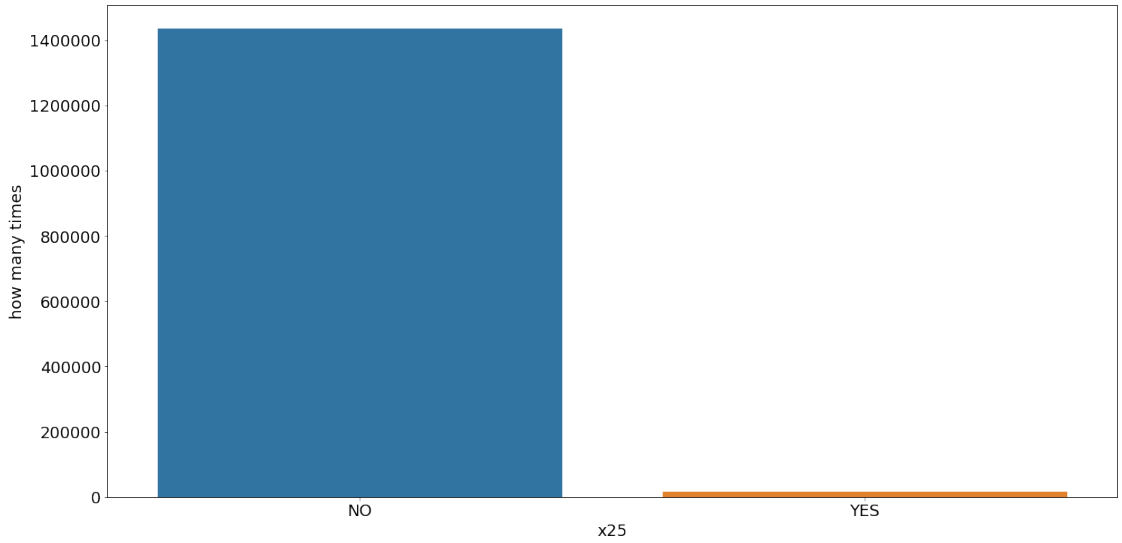
```
count     1451737
unique    2
top       NO
freq      1248943
Name: x14, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



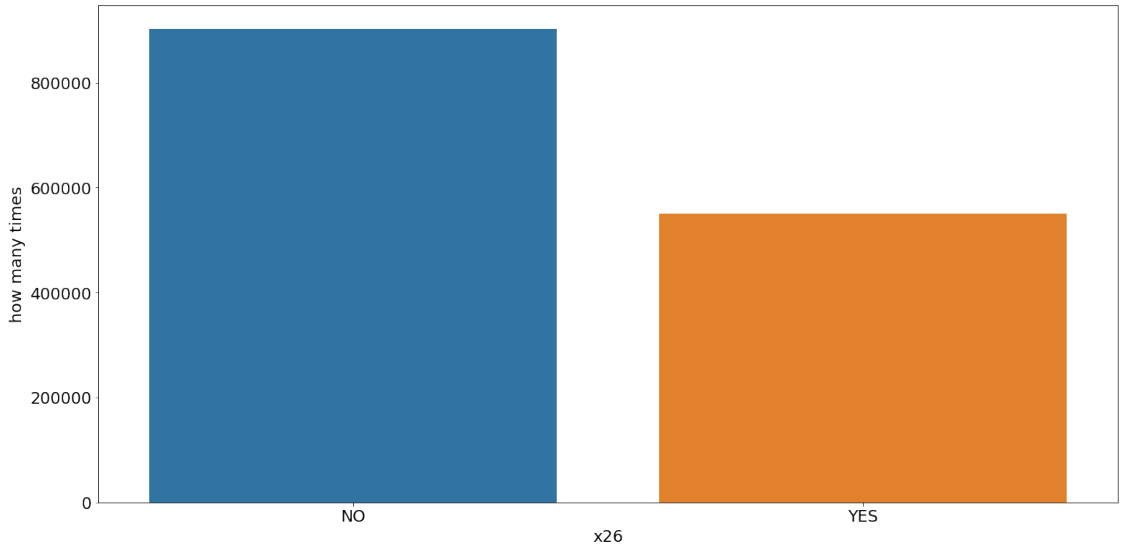

```
count     1451737
unique    2
top       NO
freq      903287
Name: x24, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



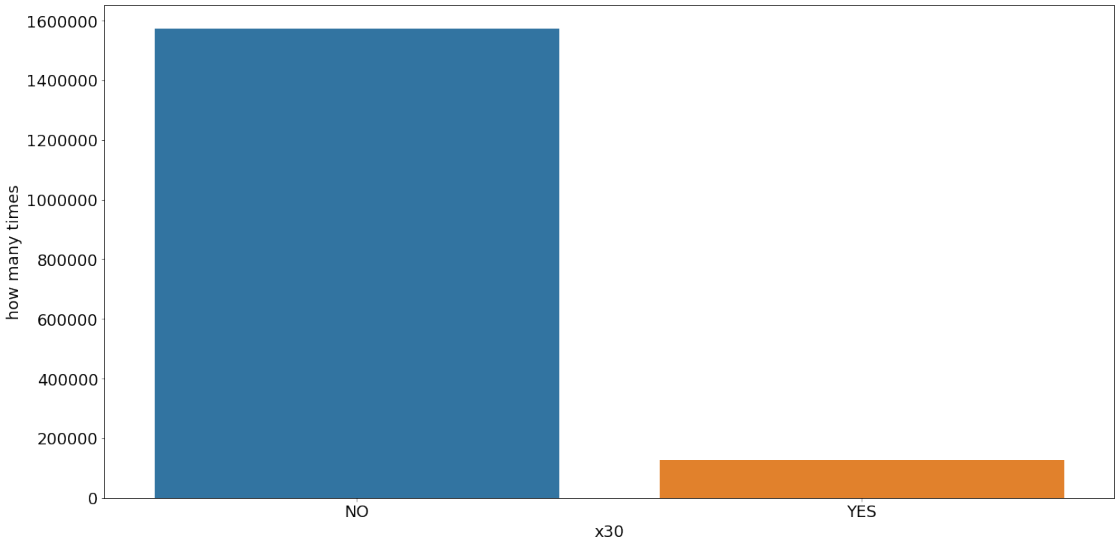

```
count      1451737
unique     2
top        NO
freq       1434703
Name: x25, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



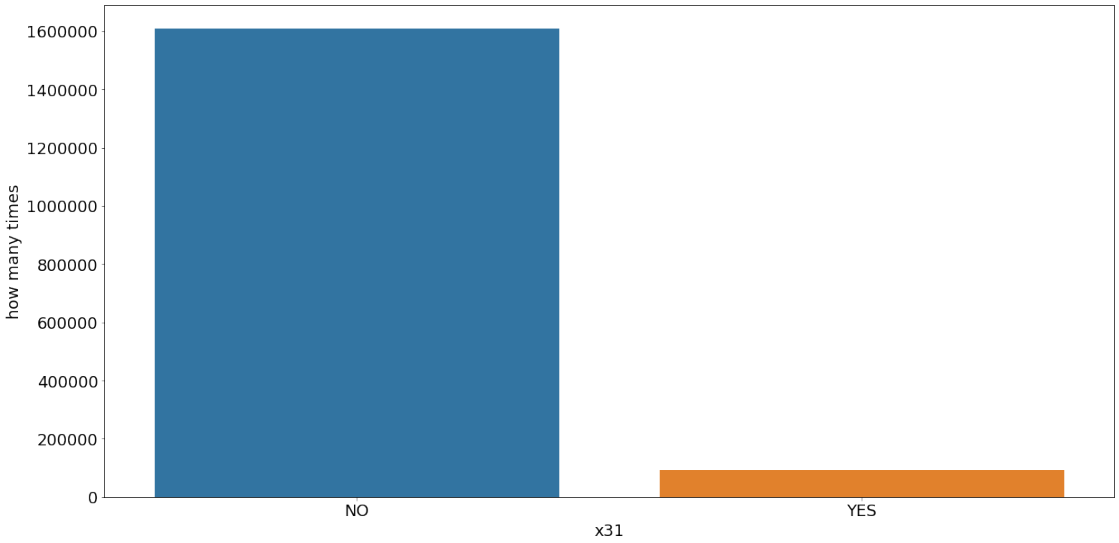
```
count     1451737
unique    2
top       NO
freq      901813
Name: x26, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



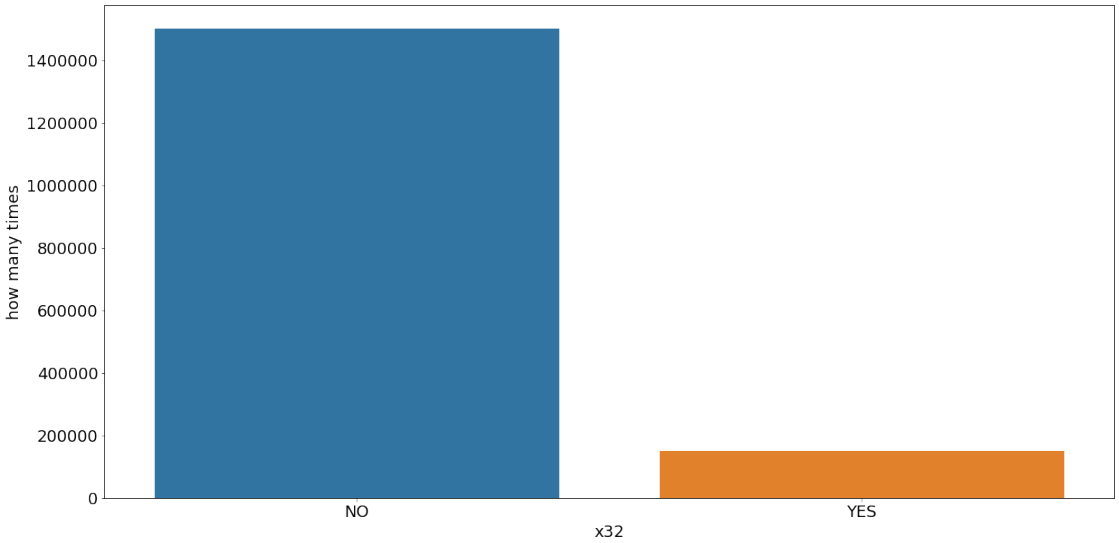

```
count     1700000
unique    2
top       NO
freq      1572446
Name: x30, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



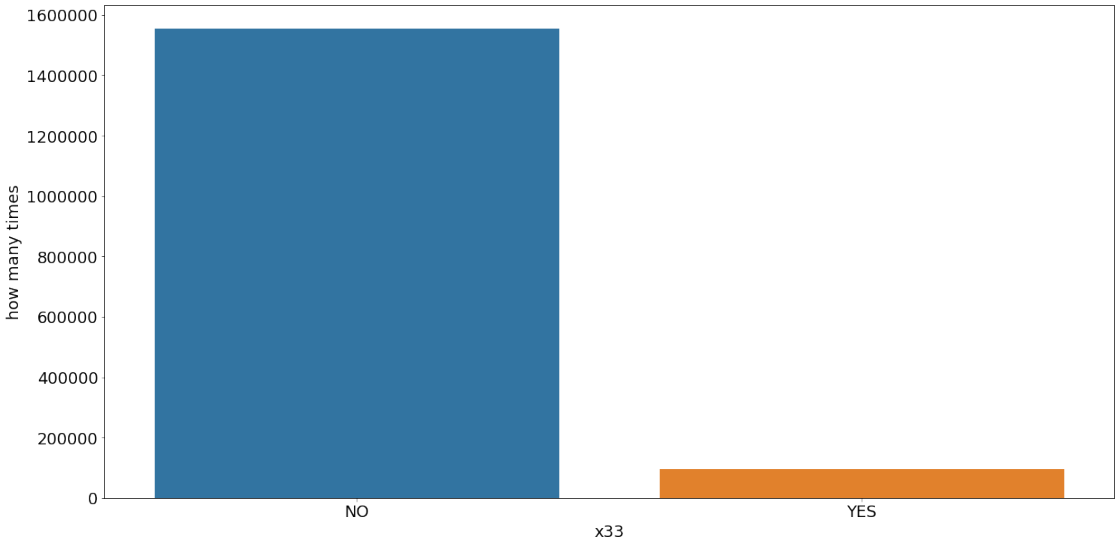

```
count      1700000
unique     2
top        NO
freq       1608242
Name: x31, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



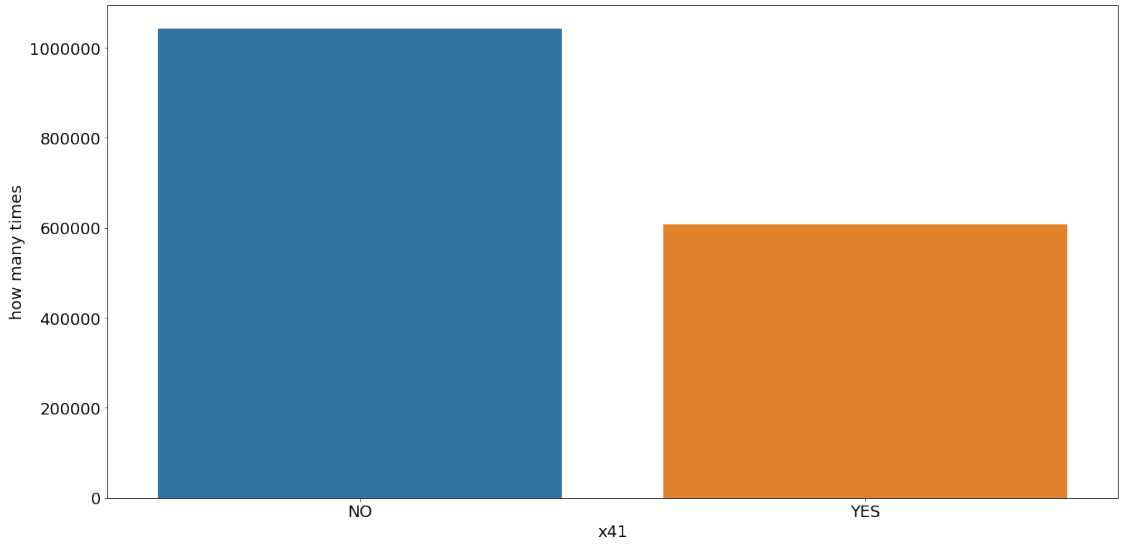
```
count     1649228
unique    2
top       NO
freq      1499709
Name: x32, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



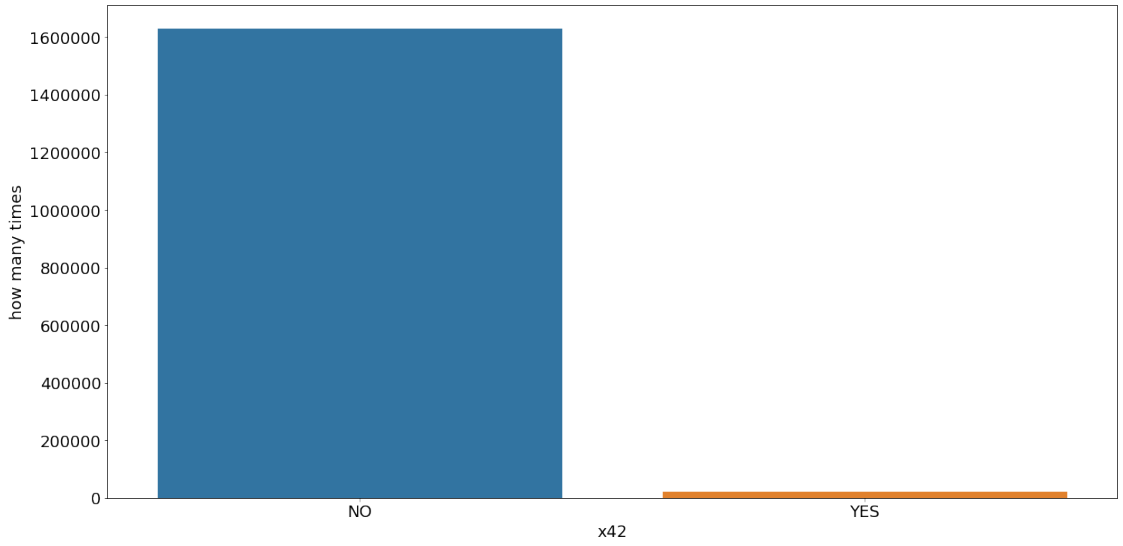

```
count     1649228
unique    2
top       NO
freq      1554186
Name: x33, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



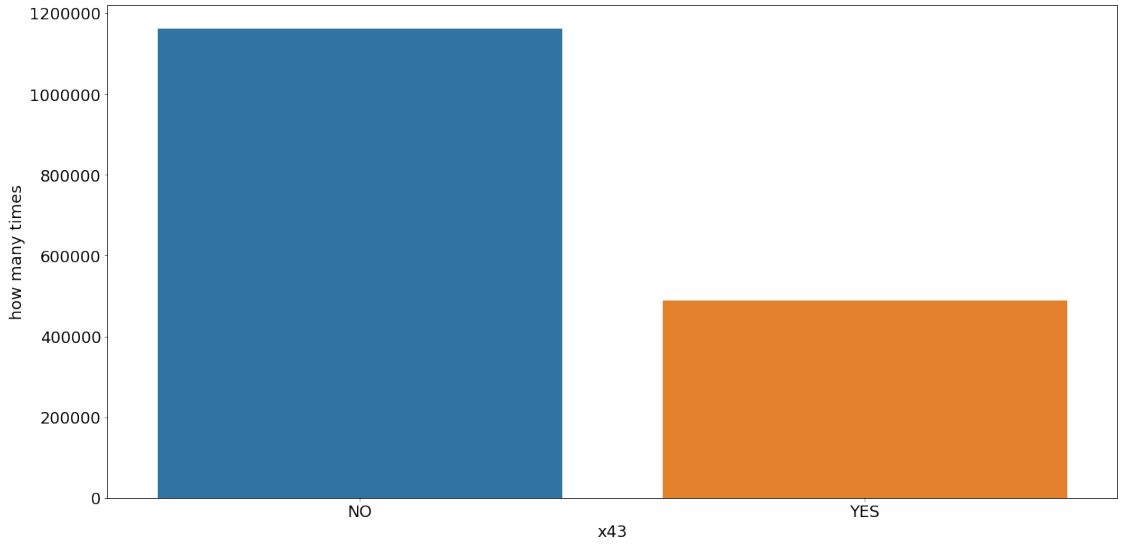

```
count     1649154
unique    2
top       NO
freq      1042290
Name: x41, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



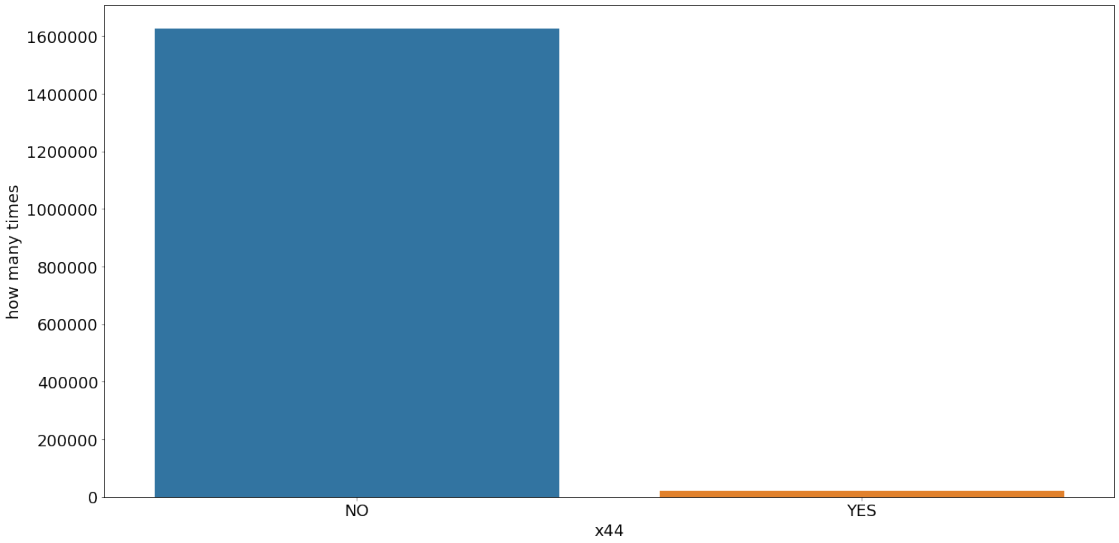
```
count     1649154
unique    2
top       NO
freq      1629261
Name: x42, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



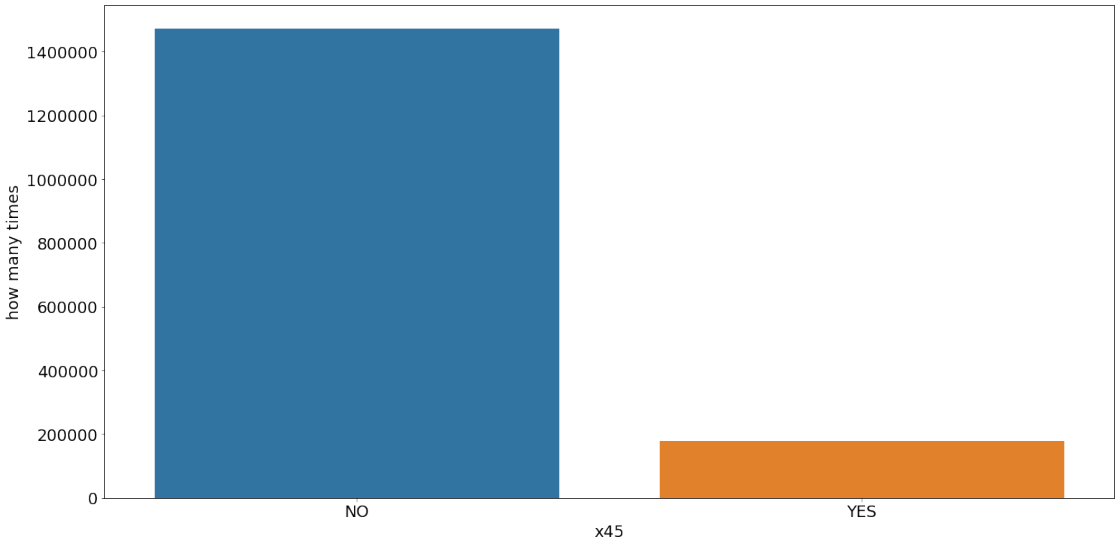

```
count     1649154
unique    2
top       NO
freq      1161460
Name: x43, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



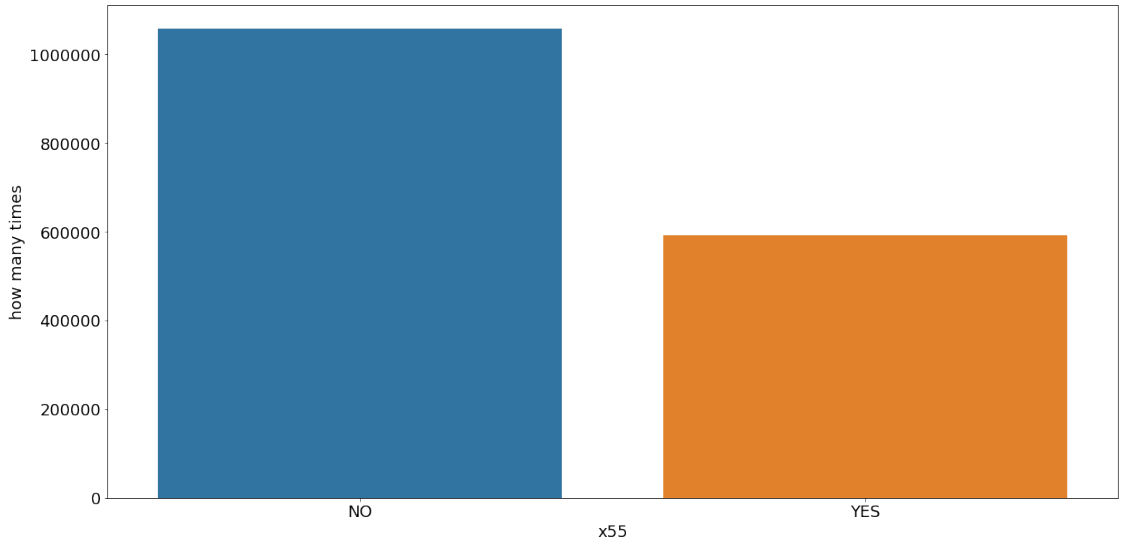

```
count      1649154
unique     2
top        NO
freq       1626229
Name: x44, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



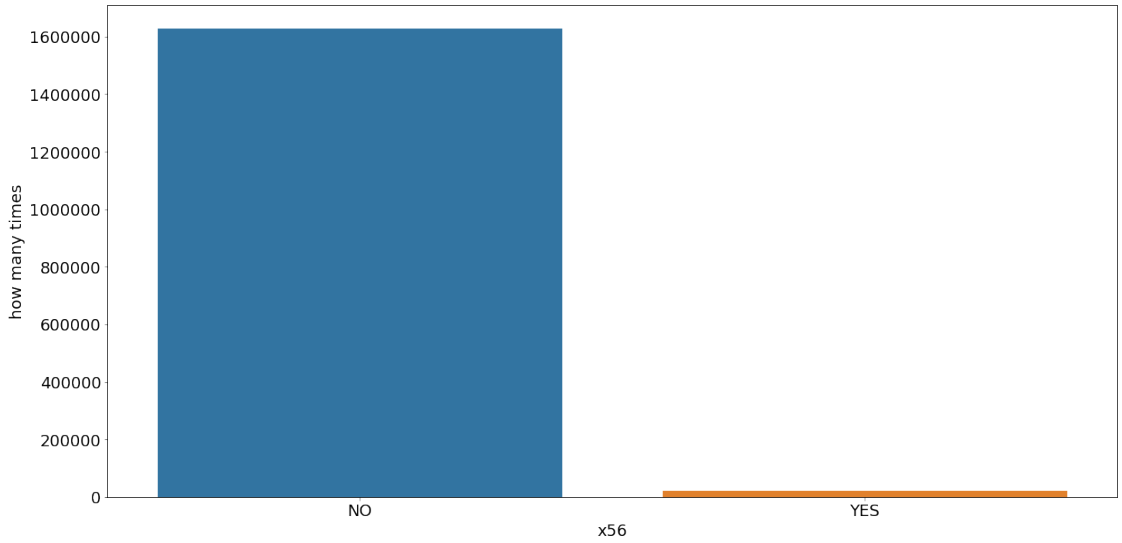
```
count     1649154
unique    2
top       NO
freq      1471789
Name: x45, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



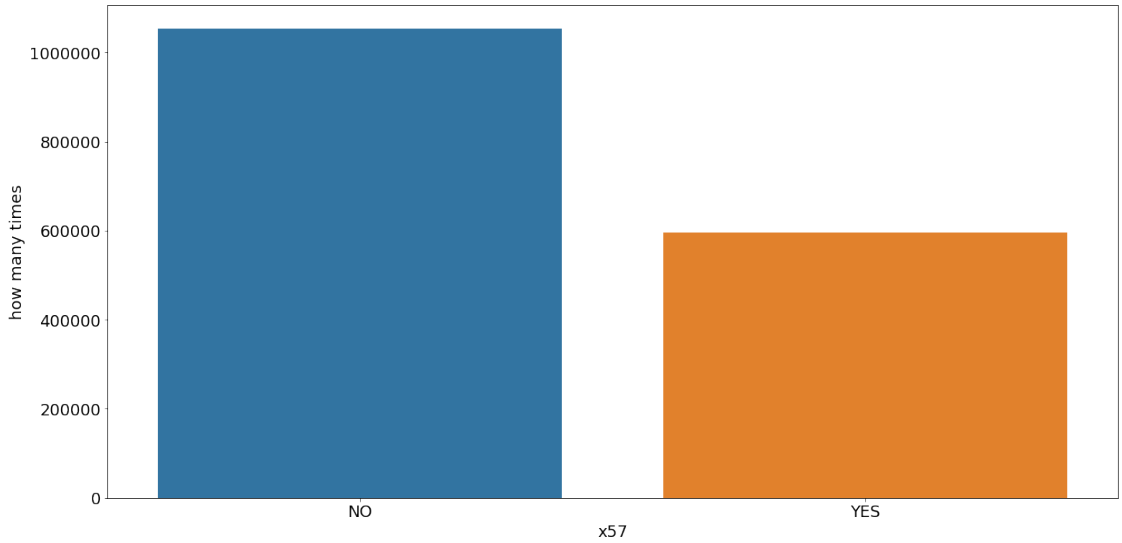

```
count     1649154
unique    2
top       NO
freq      1057557
Name: x55, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



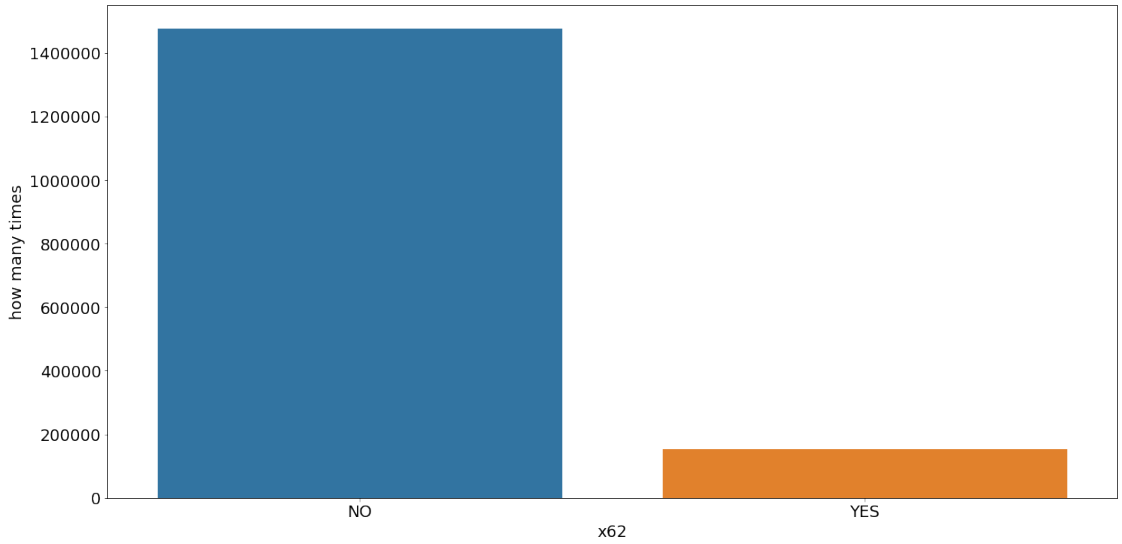

```
count      1649154
unique     2
top        NO
freq       1627610
Name: x56, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



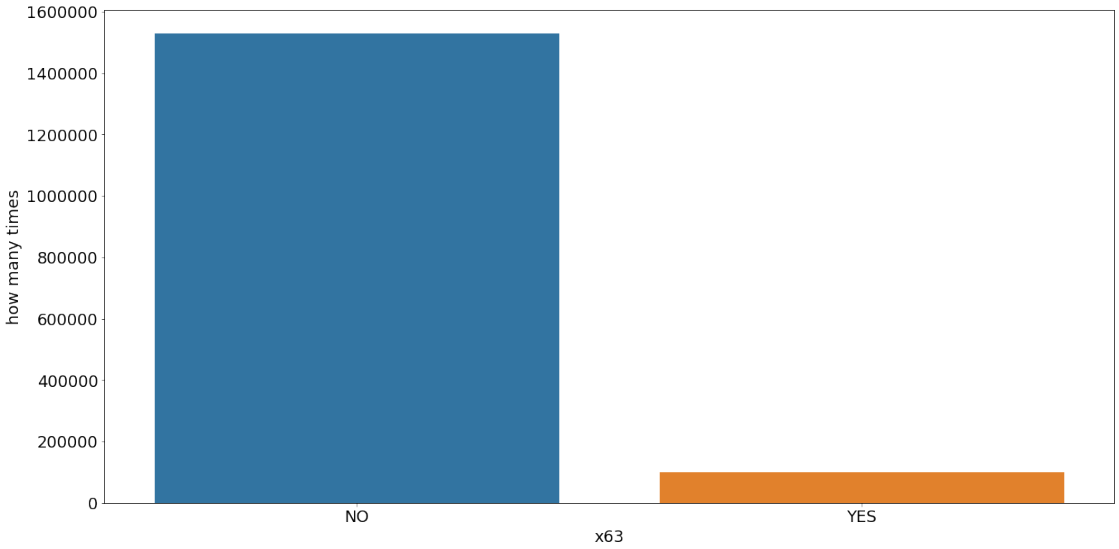
```
count     1649154
unique    2
top       NO
freq      1053102
Name: x57, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



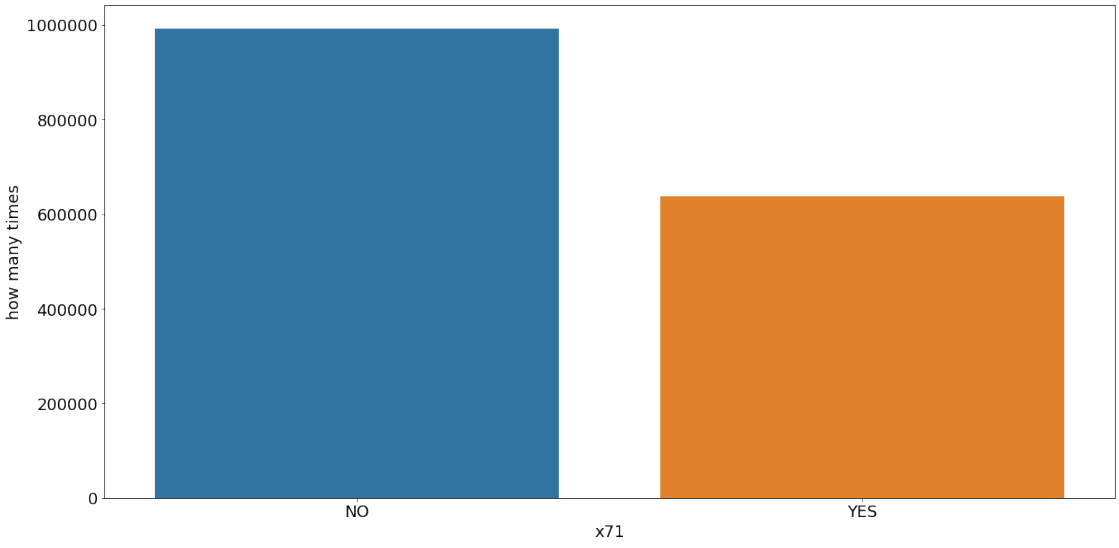

```
count     1629022
unique    2
top       NO
freq      1475852
Name: x62, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



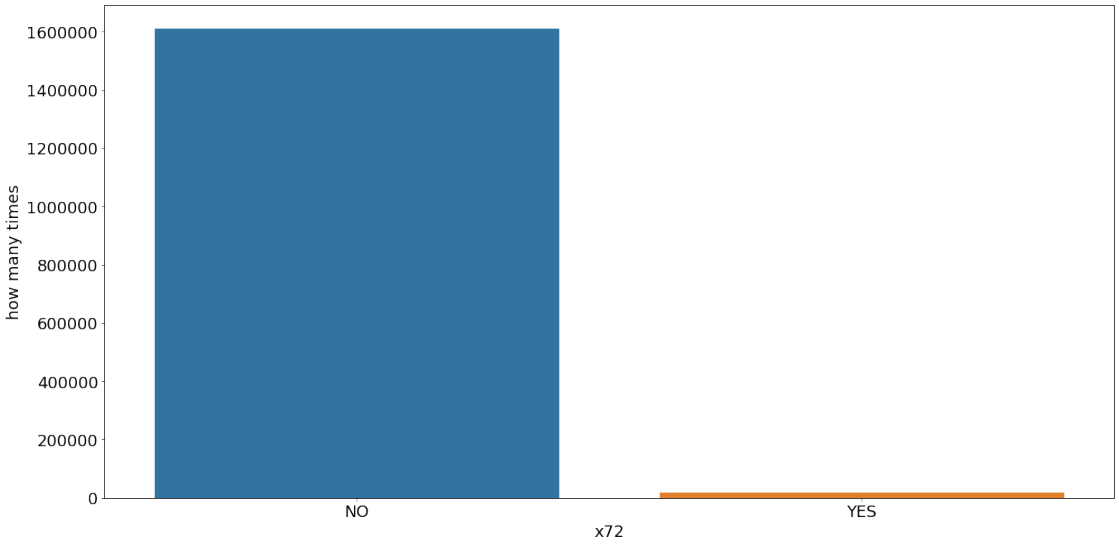

```
count      1629022
unique     2
top        NO
freq       1528719
Name: x63, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



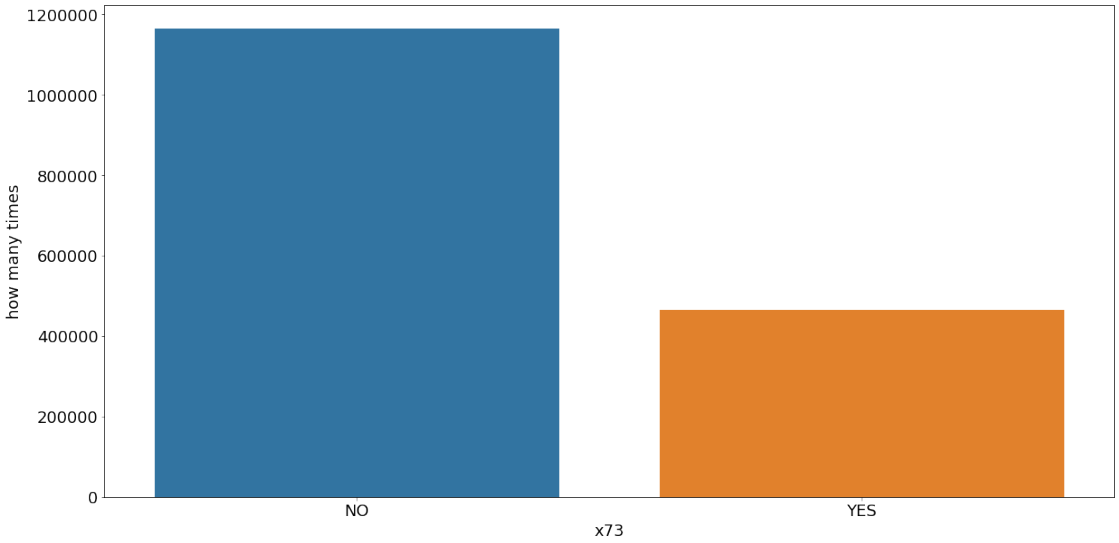
```
count     1628939
unique    2
top       NO
freq      992003
Name: x71, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



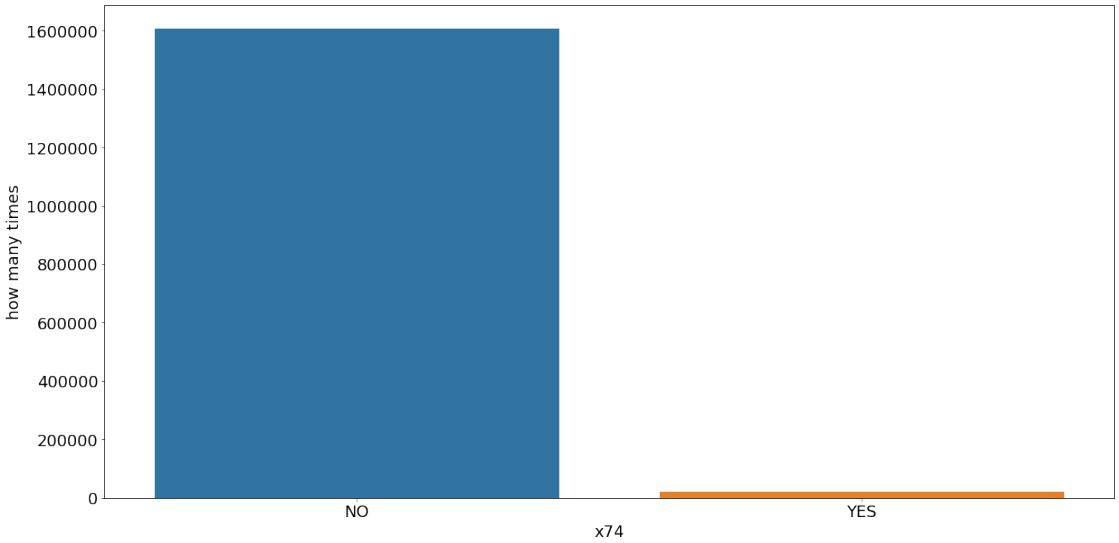

```
count     1628939
unique    2
top       NO
freq      1609920
Name: x72, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



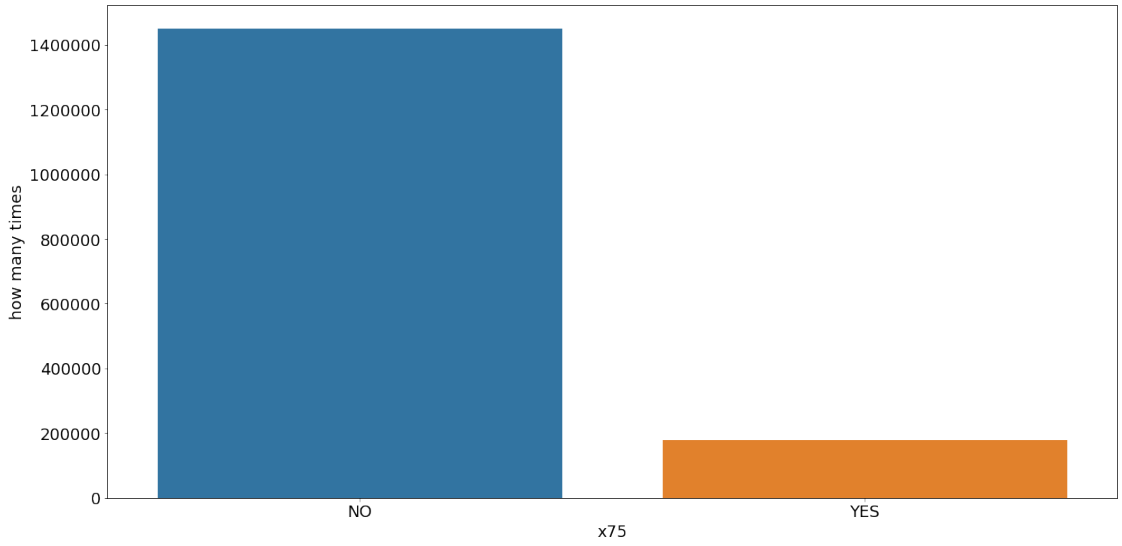

```
count      1628939
unique     2
top        NO
freq       1164110
Name: x73, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



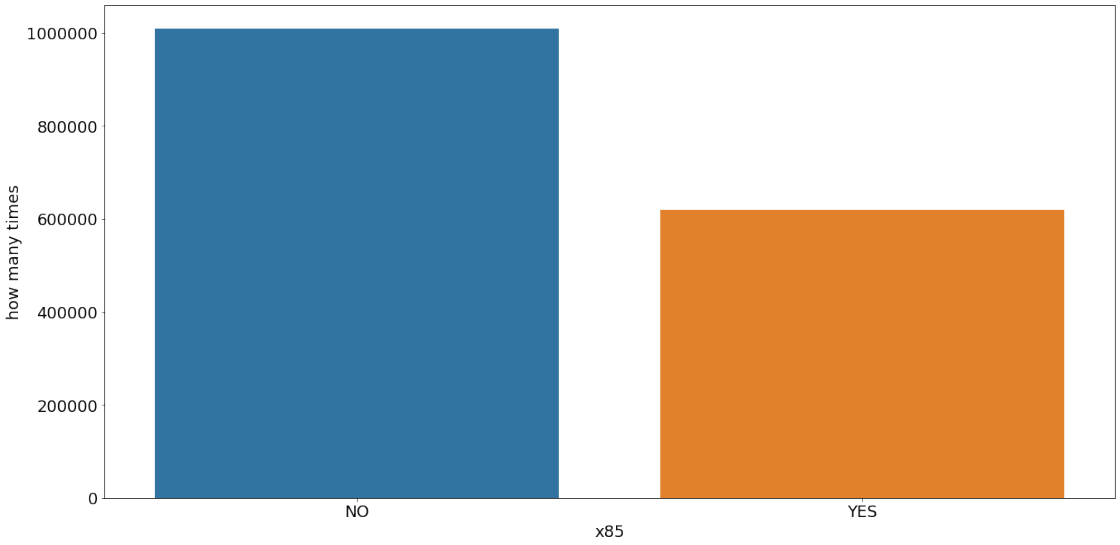
```
count     1628939
unique    2
top       NO
freq      1606349
Name: x74, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



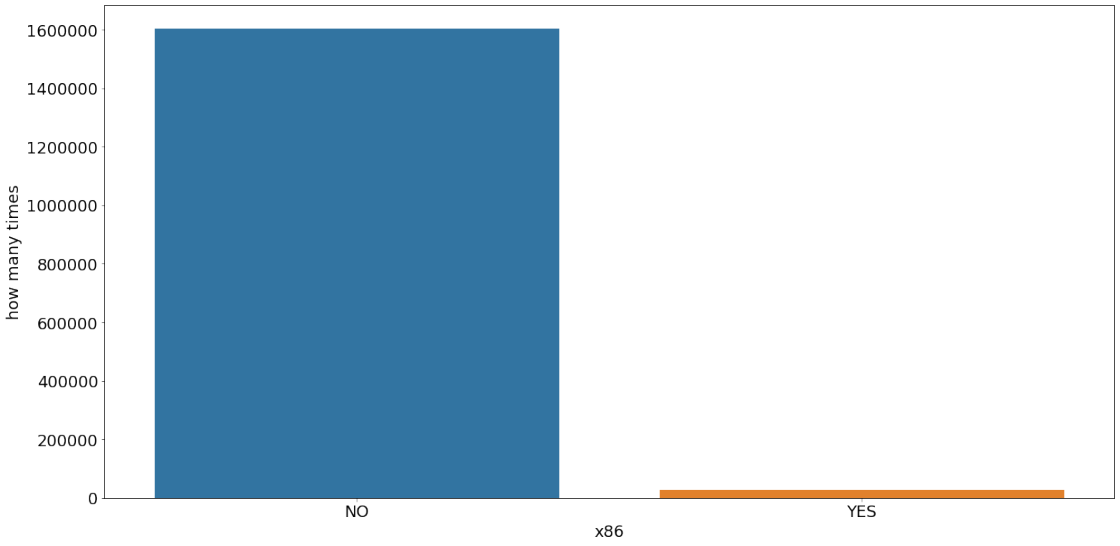

```
count     1628939
unique    2
top       NO
freq      1449499
Name: x75, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



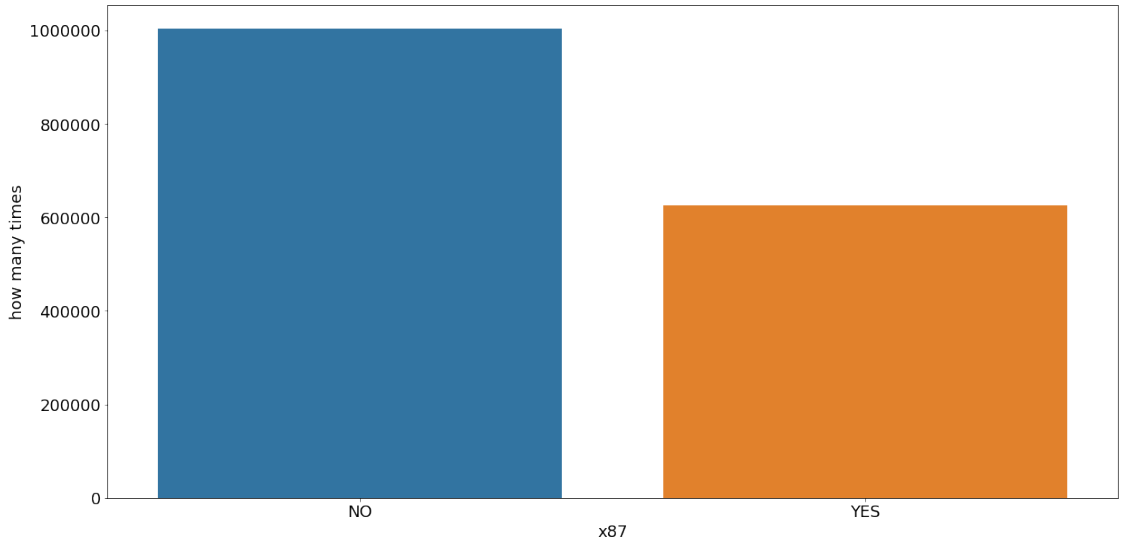

```
count     1628939
unique    2
top       NO
freq      1008828
Name: x85, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



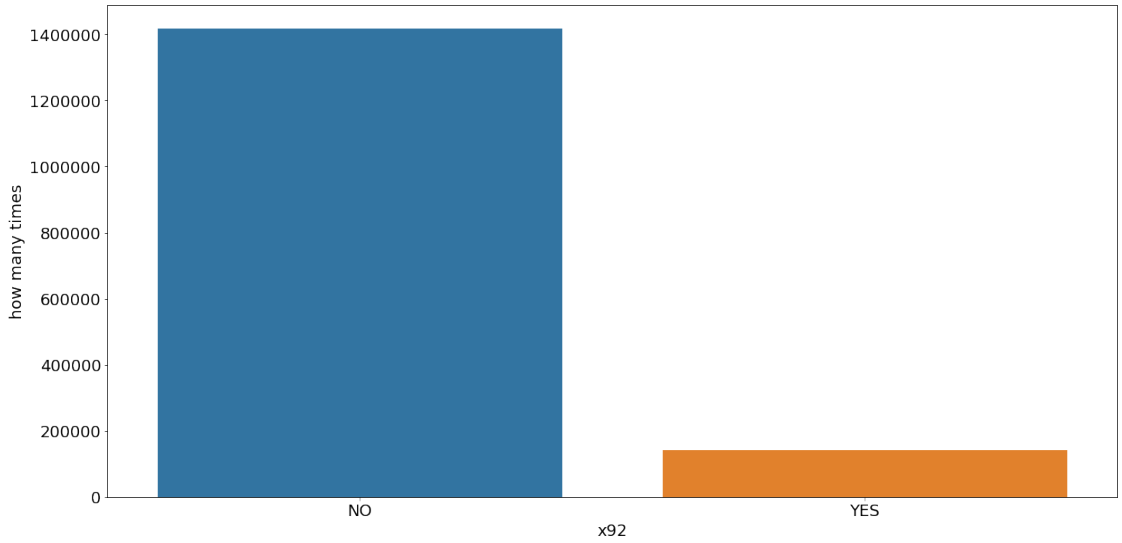
```
count     1628939
unique    2
top       NO
freq      1602879
Name: x86, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



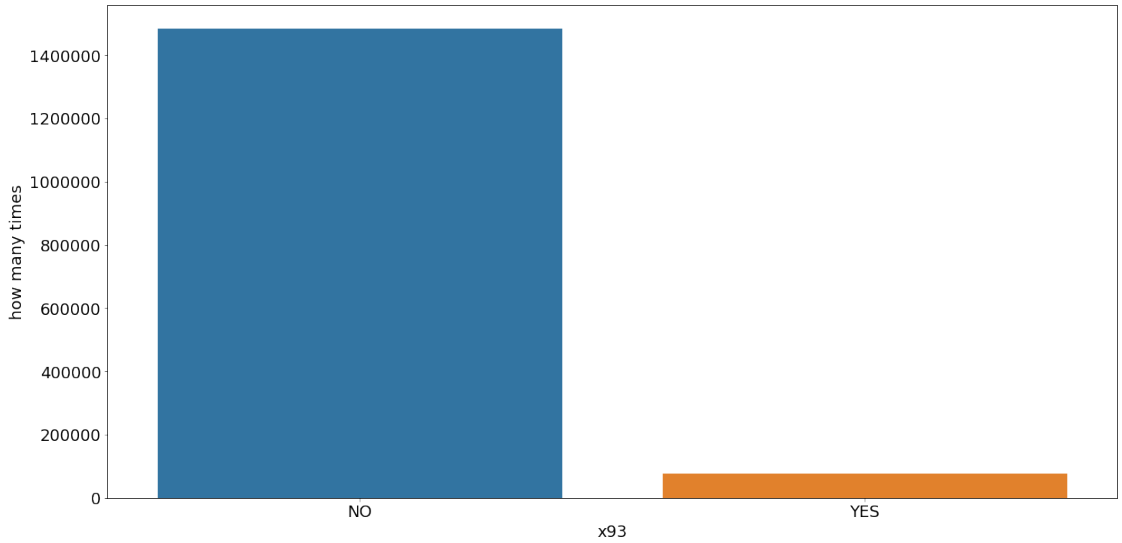

```
count     1628939
unique    2
top       NO
freq      1003502
Name: x87, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



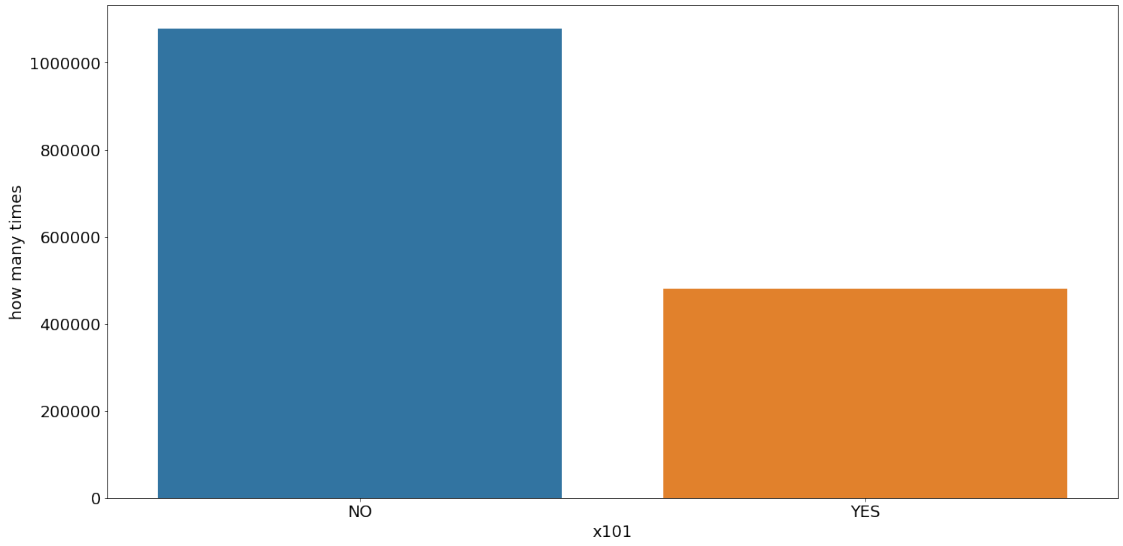

```
count      1559474
unique     2
top        NO
freq       1417147
Name: x92, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



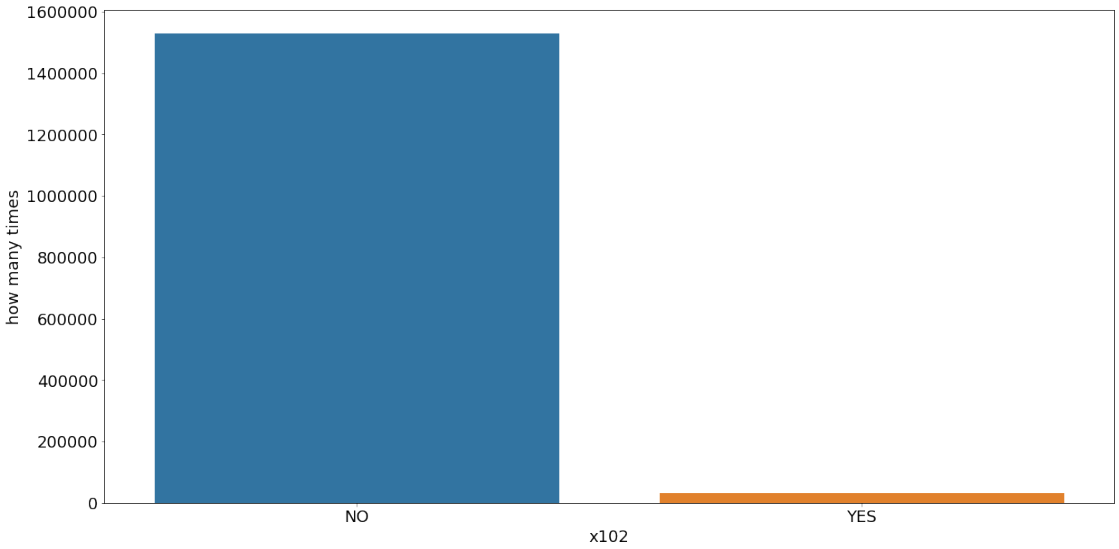
```
count     1559474
unique    2
top       NO
freq      1483269
Name: x93, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



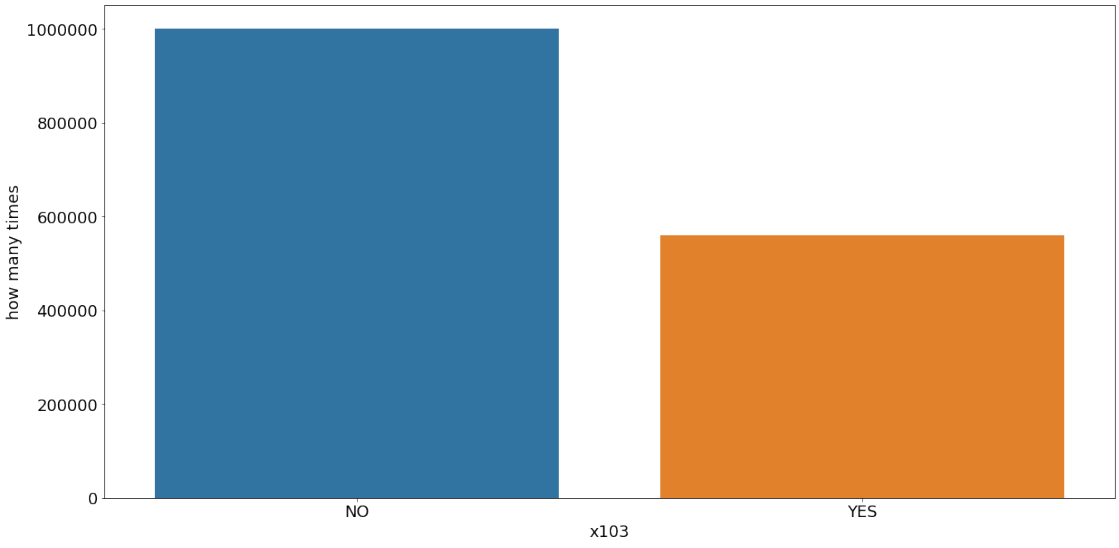

```
count     1559381
unique    2
top       NO
freq      1077855
Name: x101, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



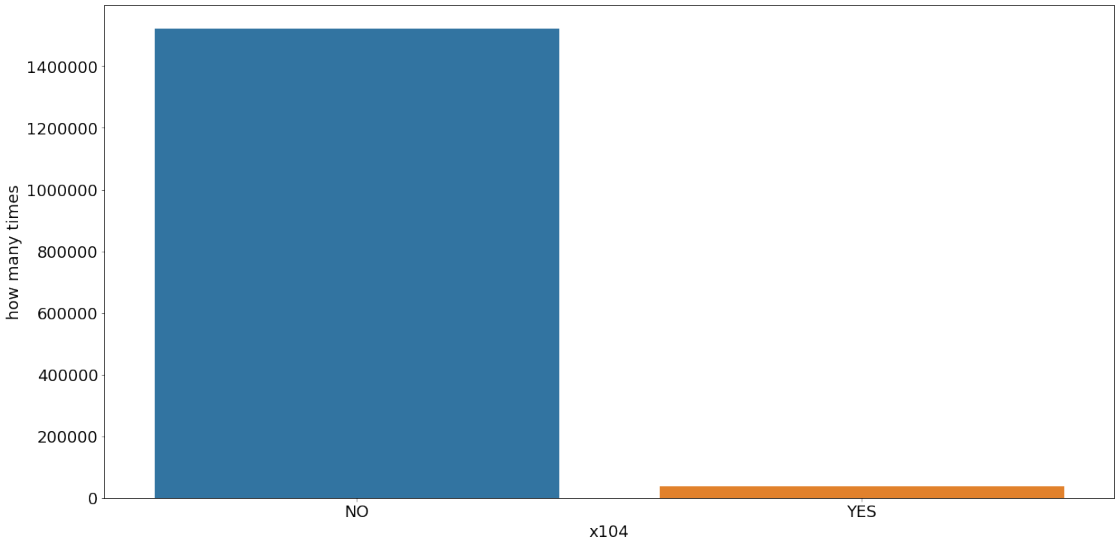

```
count      1559381
unique     2
top        NO
freq       1528452
Name: x102, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



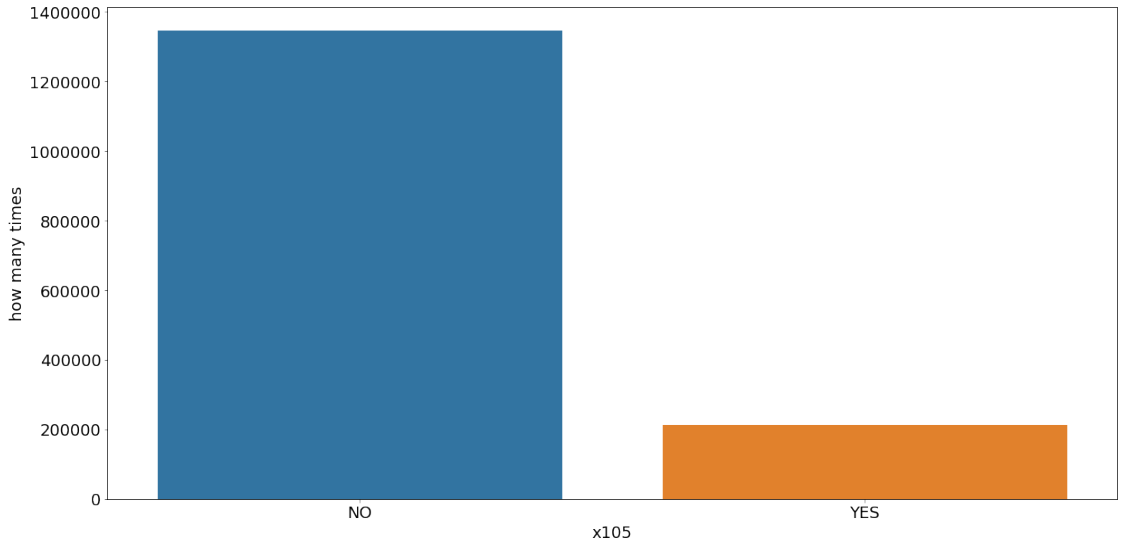
```
count     1559381
unique    2
top       NO
freq      1000298
Name: x103, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



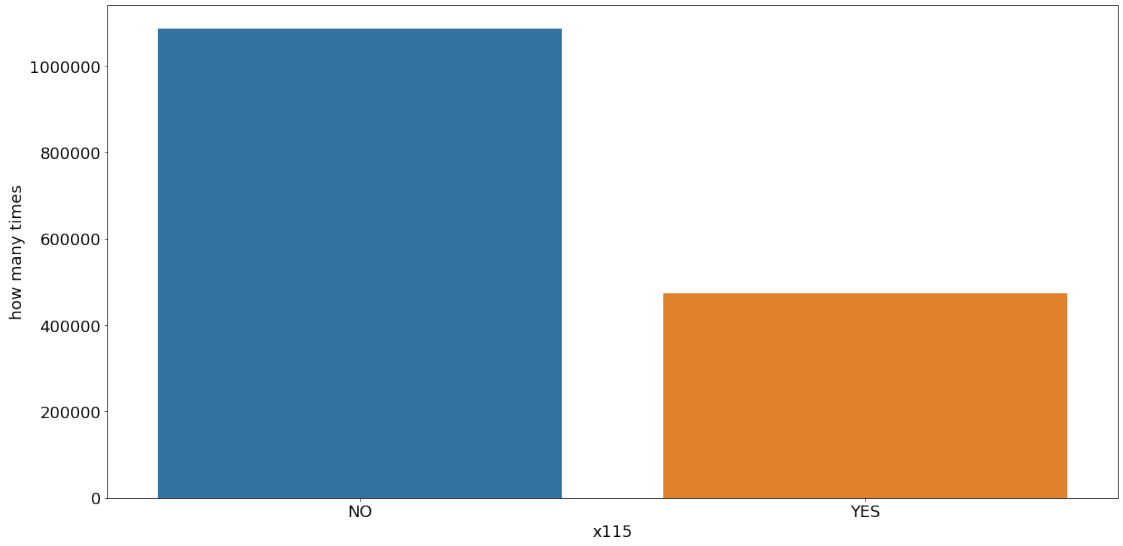

```
count     1559381
unique    2
top       NO
freq      1521380
Name: x104, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



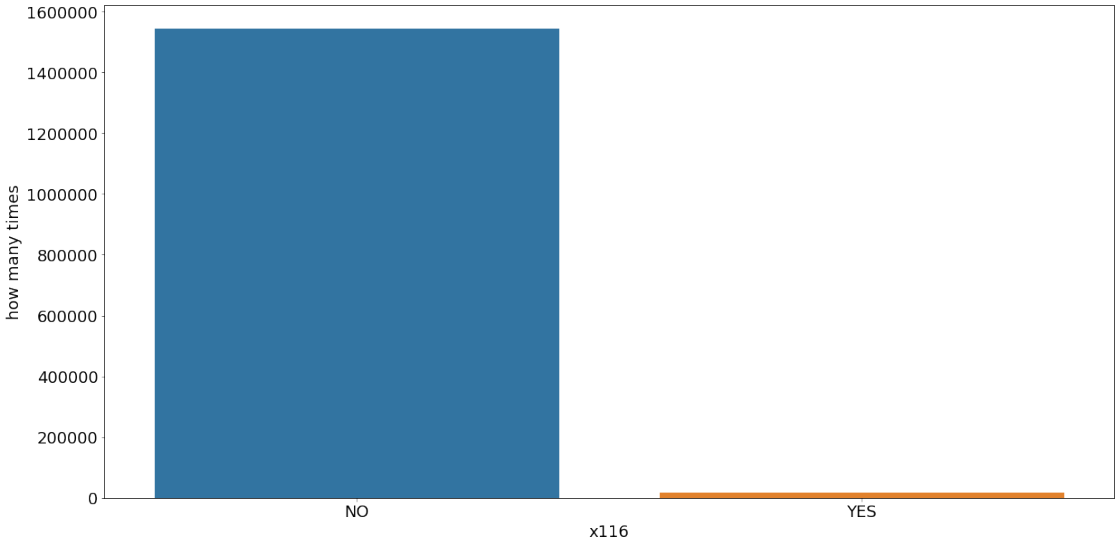

```
count      1559381
unique     2
top        NO
freq       1346098
Name: x105, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



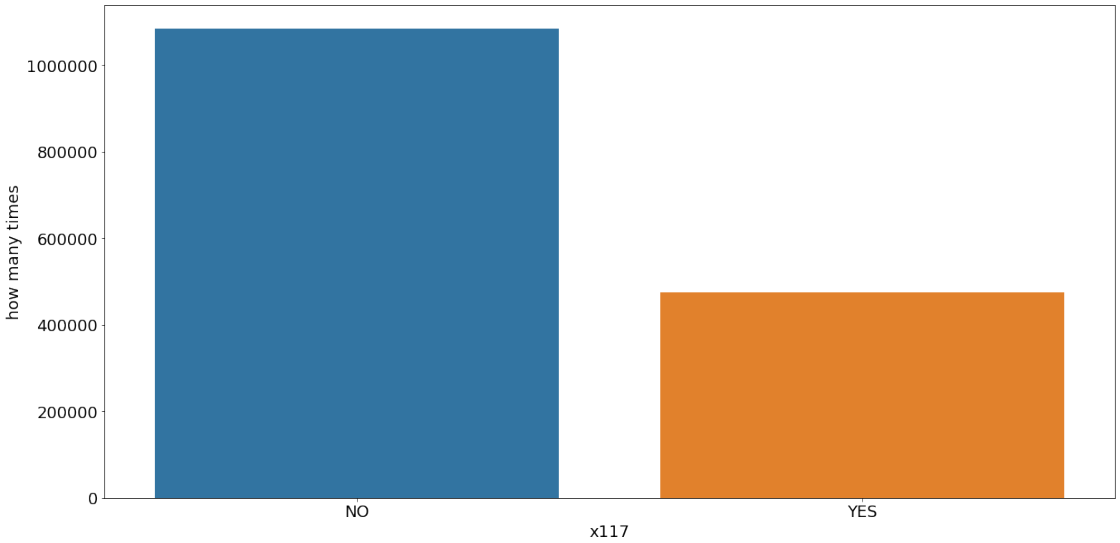
```
count     1559381
unique    2
top       NO
freq      1085899
Name: x115, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



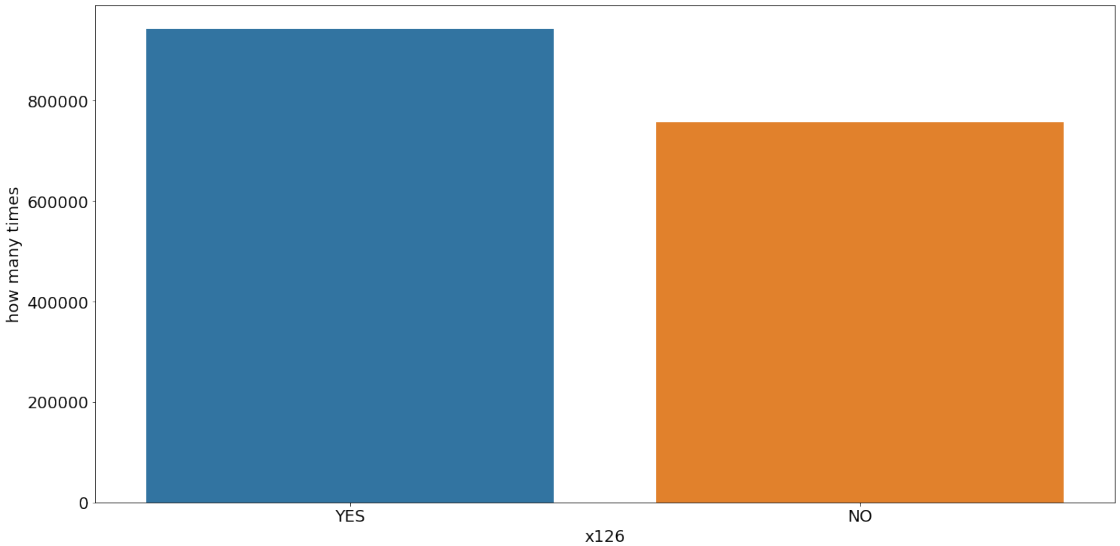

```
count     1559381
unique    2
top       NO
freq      1543503
Name: x116, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



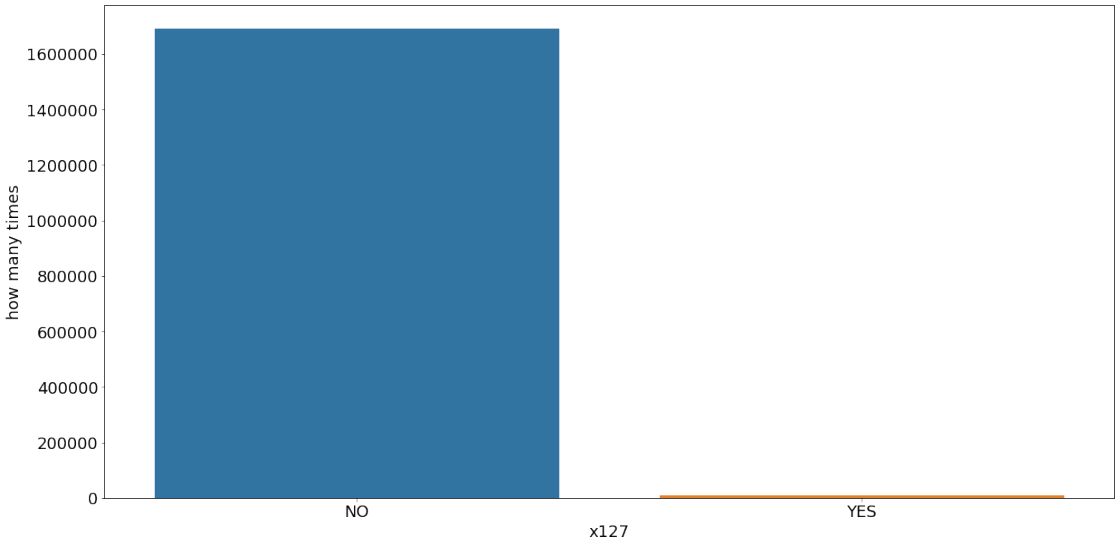

```
count      1559381
unique     2
top        NO
freq       1084300
Name: x117, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



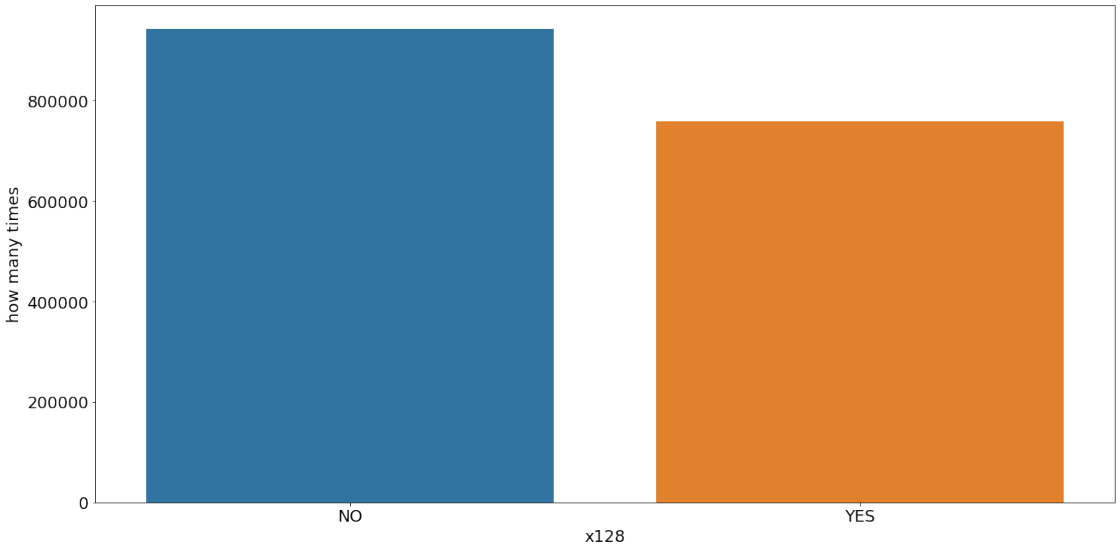
```
count     1699968
unique    2
top       YES
freq      942459
Name: x126, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



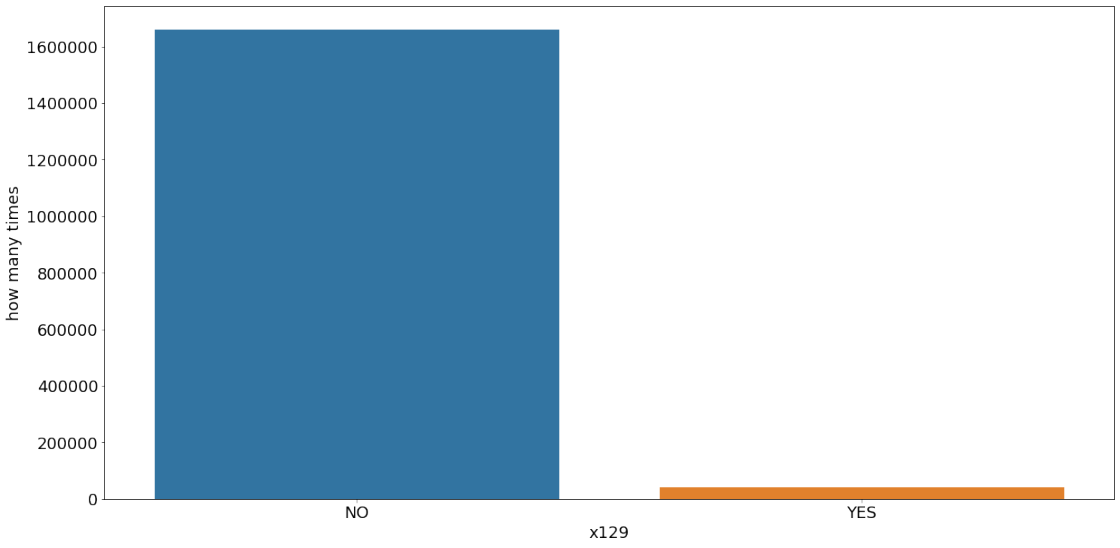

```
count     1699968
unique    2
top       NO
freq      1690819
Name: x127, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



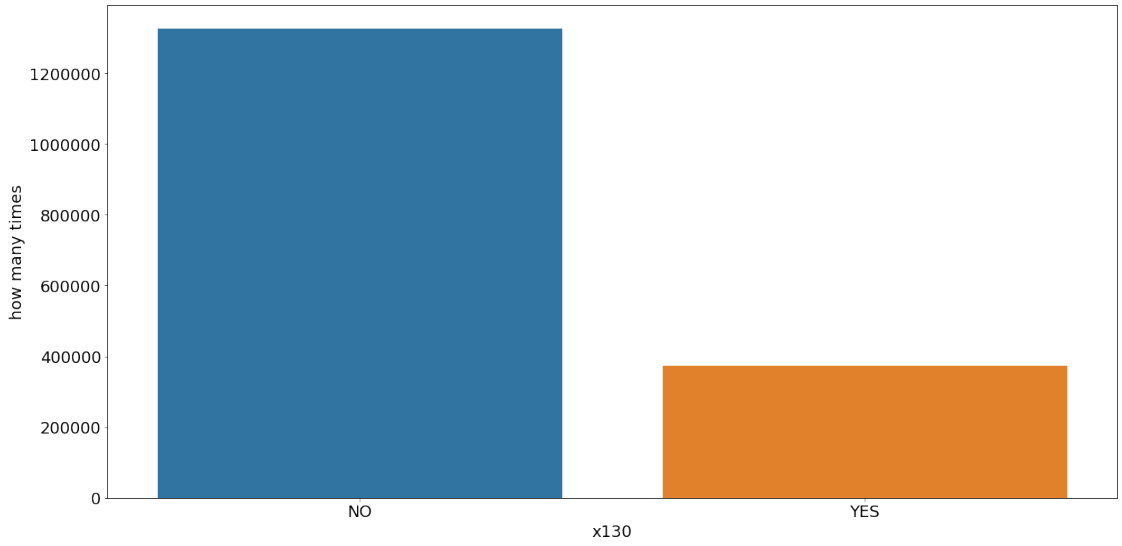

```
count      1699968
unique     2
top        NO
freq       942193
Name: x128, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



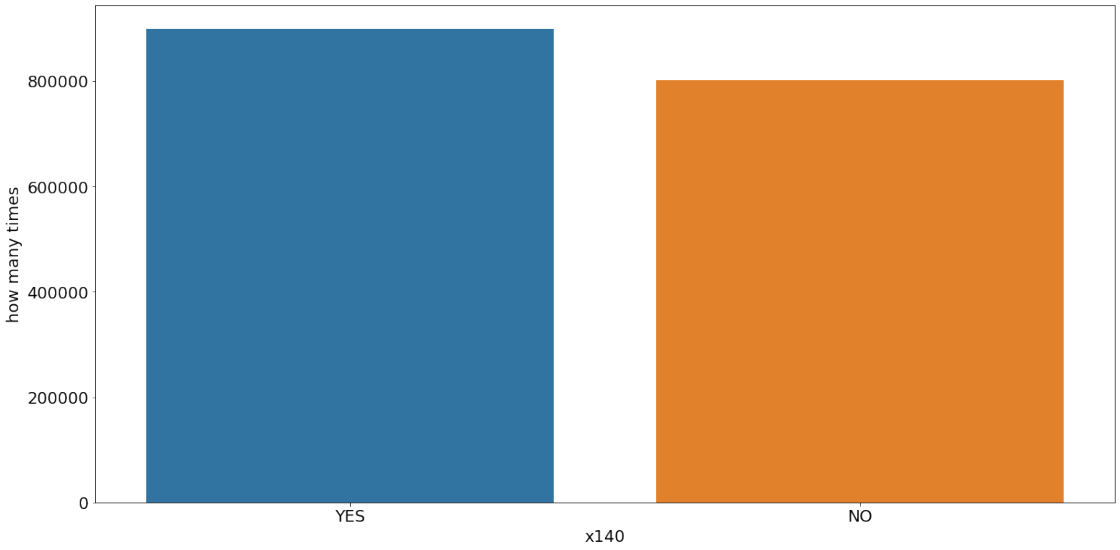
```
count     1699968
unique    2
top       NO
freq      1659101
Name: x129, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



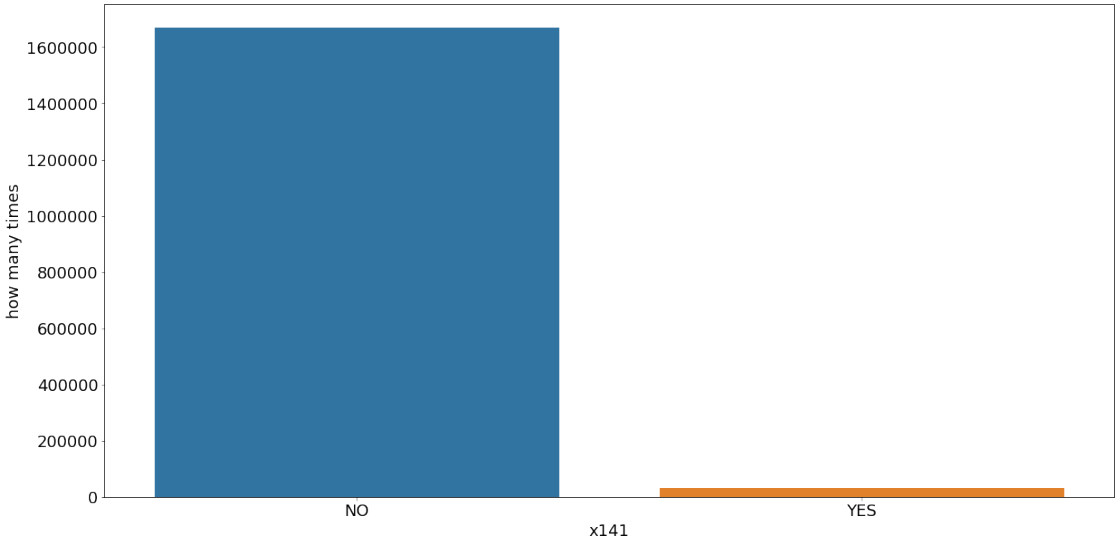

```
count     1699968
unique    2
top       NO
freq      1325460
Name: x130, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



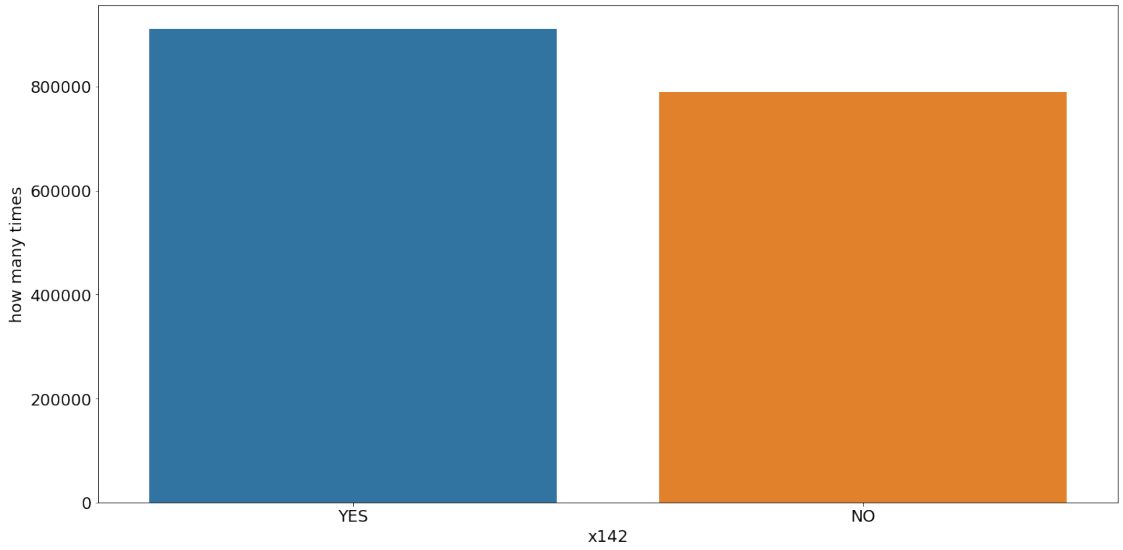

```
count      1699968
unique     2
top        YES
freq       898737
Name: x140, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```



```
count     1699968
unique    2
top       NO
freq      1668833
Name: x141, dtype: object
```

```
<Figure size 432x288 with 0 Axes>
```




```
count     1699968
unique    2
top       YES
freq      910356
Name: x142, dtype: object
```

The majority of the boolean variables have a greater quantity of NO than YES, only on x126, x140,x142 the YES values are greater than NO values. This visualization don't give us too much information as we don't know which any boolean feature means.

1.3.3 Algorithms and Techniques

I intend to use three different algorithms to beat the benchmark: Random Forest Classifier, Nearest Neighbors, and Multilayer Perceptrons.

The first one, Random Forest Classifier is a bagging method, where they create multiples independents trees on subsets of the features and then average the result of each tree. The result of the average of all tree is a better estimator than a single tree because the variance is reduced, reducing the overfitting problem.

The second one is Linear SVC, a Support Vector Machine method, which can be effective in high dimensional spaces, in memory usage, when the number of dimensions is greater than the number of samples.

The last one, Multi Layer Perceptron or Neural Networks, has the capability to learn non-linear function which can be advantageous in this project, but this kind of model has the disadvantage to be sensitive to feature scaling, requires a lot of tuning in the hyperparameters and can be stuck on a local minimum depends on the weight initializations.

1.3.4 Benchmark

In this competition, the organizers give us fours different benchmarks scores: * The all zeros benchmark, where every label answer is zero; * The random benchmark, where every label was given a random value; * The halves benchmark, where every label was given a value of 0.5; * And finally the Tradeshift Baseline Benchmark.

The score of the four benchmarks is find below, in this evaluation metric lower is better:

Benchmark	Score
TS Baseline	0.0150548
All Halves	0.6931471
Random	1.0122952
All Zeros	1.1706929

```
In [18]: # testing the score function
         y_true = [1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000, 0.0000, 1.0000, 0.0000, 0.0
         y_pred = [0.9000, 0.1000, 0.0000, 0.3000, 0.0300, 0.7000, 0.2000, 0.8500, 0.1900, 0.0

         log_loss(y_true, y_pred)

Out[18]: 0.1554686986370972

In [19]: #testing the score function on all halves benchmark
         y_pred = np.full((1700000, 33), 0.5, dtype=np.float64).flatten()
         y_true = np.array(train_labels.values)[:, 1:].astype(np.float64).flatten()

         log_loss(y_true, y_pred)

Out[19]: 0.69314718056002567
```

1.4 III. Methodology

1.4.1 Data Preprocessing

Memory efficient In this part I will explore some transformation on the dataset for memory efficiency, like the transformation of the YES/NO features to 0/1.

```
In [27]: meta_path = "../working/meta.pkl"

         if not os.path.isfile(meta_path) :
             pickle.dump(meta, open(meta_path, "wb"))
```

```
meta = pickle.load(open(meta_path, "rb"))

int_features = meta[(meta.dtype == 'int64') & (meta.role != 'id')].index
float_features = meta[(meta.category == 'numerical') & (meta.dtype == 'float64')].inde
bool_vars = meta[(meta.category == 'boolean')].index
content_features = meta[(meta.category == 'content')].index
```

As this dataset is too big for the memory I have available, I will work with only one fifth of the dataset.

```
In [6]: ids_samples = [hash(i_id) % 5 == 0 for i_id in train_features['id']]

In [7]: train_features = train_features[ids_samples]
        train_features.shape

Out[7]: (340000, 146)

In [8]: train_labels = train_labels[ids_samples]
        train_labels.shape

Out[8]: (340000, 34)

In [9]: # drop the empty label
        train_labels.drop(labels=['y14'], axis="columns", inplace=True)
        train_labels.shape

Out[9]: (340000, 33)

In [10]: train_labels.to_pickle('../working/1_test_reduced.pkl')

In [11]: train_features = pre.tranform_bool_df(train_features, bool_vars)
         train_features.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 340000 entries, 4 to 1699999
Columns: 146 entries, id to x145
dtypes: float32(50), float64(55), int64(31), object(10)
memory usage: 316.5+ MB


In [12]: train_features[bool_vars].isnull().any().any()

Out[12]: False
```

This transformation decreases the memory to in ~320 MB comparing to 1.8+ GB of the original dataset.

Encoding the content As the algorithms accept only floats, ints and booleans as input, I decide to tranform all the contents feature to boolean features where the column represent the combo feature plus the hash and the value is 1 if the pair exists or 0 if not, as this has a high memory comsuption I will go for a sparse matrix type of collumn when use get_dummies on pandas.

```
In [13]: train_features = pre.transform_content_dummy(train_features, content_features)
```

```
Index(['x3', 'x4', 'x34', 'x35', 'x61', 'x64', 'x65', 'x91', 'x94', 'x95'], dtype='object')
Index(['x1', 'x10', 'x100', 'x101', 'x102', 'x103', 'x104', 'x105', 'x106',
       'x107',
       ...
       'x88', 'x89', 'x9', 'x90', 'x92', 'x93', 'x96', 'x97', 'x98', 'x99'],
      dtype='object', length=135)
```

```
In [14]: train_features = pre.transform_sparse(train_features, content_features)

In [15]: vec = pd.read_pickle("../working/content_dicvector.pkl")

In [16]: np.nan in vec.feature_names_

Out[16]: False

In [17]: train_features

Out[17]: <340000x510290 sparse matrix of type '<class 'numpy.float64'>'
                with 3400000 stored elements in Compressed Sparse Row format>
```

After the transformation the dataset goes from 10 content input columns to 510_290.

Scaling In this section, will scale the dataset to get a better result on the PCA as they need similar scales of measurement.

```
In [18]: train_features = pd.read_pickle("../working/3_train_numerical.pkl")
         train_features = pre.transform_scale_float(train_features, float_features)
         train_features[float_features].head()
```

```
Out[18]:          x5        x6        x7        x8        x9       x16       x19       x20
         4   0.872345 -0.419096  0.589557 -0.520604 -0.234012  0.351686  0.422363  0.129417  0
         9   0.415617  4.555282 -0.014624  2.968602  1.270081  0.977167  0.453683  0.129417  0
         14  0.872345 -0.419096  0.589557 -0.520604 -1.186334 -1.021071  0.409261  0.129417  0
         19  0.694659  1.556248 -0.232860  3.379537  1.582892 -0.036996  0.409378  0.129417  0
         24  0.241649  0.686067 -1.226389  2.698874  1.056054 -0.790015  0.479274  0.129417  0
```

```
In [19]: train_features = pre.transform_scale_int(train_features, int_features)
         train_features[int_features].head()
```

```
Out[19]:          x15       x17       x18       x22       x23       x27       x46       x48
         4  -0.576965 -0.107787  0.406107 -0.595154 -0.647305 -0.184162 -0.209320 -0.427920 -0
         9  -0.576965  0.108376 -0.433412  1.362158  0.947245  0.041473 -0.595584 -0.427920 -0
         14  0.541195 -0.323949 -0.853171 -0.595154 -0.647305 -0.409797 -0.595584 -0.427920 -0
         19 -0.465149  0.756863  0.546026  1.361012  0.943949  0.267108 -0.595584  0.110422  0
         24 -0.241517 -0.540112 -0.293492  1.362158  0.947245 -0.184162  0.949472 -0.320252 -0
```

PCA As the previous PCA run show us that we have the width/height features with a lot of linear correlation so will apply in this columns.

```
In [20]: # wh -> widht/height
         wh_features = meta[(meta.category == 'height') | (meta.category == 'width')].index
         train_features = pre.transform_pca(train_features, wh_features)
         train_features.head()
```

Out[20]:

	x1	x10	x100	x101	x102	x103	x104	x105	x106	x107	x108	
4	0.0	0.0	-0.357775	0.0	0.0	0.0	0.0	0.0	-0.685033	-1.140945	0.704878	0
9	0.0	1.0	1.231119	1.0	0.0	0.0	0.0	0.0	-0.585210	1.186468	-0.414582	-0
14	0.0	0.0	-1.366631	0.0	0.0	1.0	0.0	0.0	-0.585210	-1.145154	-0.414582	-0
19	0.0	0.0	1.565492	0.0	0.0	0.0	0.0	0.0	-0.585210	0.020752	0.257094	0
24	0.0	0.0	1.007379	0.0	1.0	1.0	0.0	1.0	1.111795	-0.837408	-0.862367	-0

Now, I will join the datasets together to feed the machine learning algorithms

```
In [21]: train_wh_pca_path = "../working/9_train_wh_pca.pkl"
         numerical_train_features = pd.read_pickle(train_wh_pca_path)

         # on the join all the features requires to be the same type
         numerical_train_features = np.array(numerical_train_features).astype(np.float32)
```

```
In [22]: train_only_content_encoding = "../working/4_train_only_content_encoding.pkl"
         content_train_features = pd.read_pickle(train_only_content_encoding)
         content_train_features = content_train_features.astype(np.float32)
         content_train_features.data = np.nan_to_num(content_train_features.data)
```

```
In [23]: content_train_features.shape
```

```
Out[23]: (340000, 510290)
```

```
In [24]: numerical_train_features.shape
```

```
Out[24]: (340000, 130)
```

```
In [25]: train_features = sparse.csr_matrix(sparse.hstack([sparse.coo_matrix(numerical_train_fe
```

```
In [26]: train_features
```

```
Out[26]: <340000x510420 sparse matrix of type '<class 'numpy.float32'>'
                with 33597107 stored elements in Compressed Sparse Row format>
```

```
In [27]: pickle.dump(train_features, open('../working/11_train_pre_processed.pkl', "wb"))
```

Split the dataset on training and testing, saving to rerun this algorithms.

```
In [30]: train_labels = pd.read_pickle('../working/1_test_reduced.pkl')
```

```
In [31]: train_features = pd.read_pickle('../working/11_train_pre_processed.pkl')

         X_train, X_test, y_train, y_test = train_test_split(train_features, train_labels, test

         # drop id
         y_train = np.array(y_train)[:,1:]
         y_test = np.array(y_test)[:,1:]

         x_train_path = "../working/12_train_x.pkl"
         x_test_path = "../working/13_test_x.pkl"
         y_train_path = "../working/14_train_y.pkl"
         y_test_path = "../working/15_test_y.pkl"

         pre.save_dataset(X_train, x_train_path)
         pre.save_dataset(X_test, x_test_path)
         pre.save_dataset(y_train, y_train_path)
         pre.save_dataset(y_test, y_test_path)
```

1.4.2 Implementation

First of all, I will try the algorithms with only the numerical features to see where we are compared with the benchmark. And will have to split the training dataset in 2 parts to training and validation.

Numerical part of Dataset

```
In [52]: from sklearn.metrics import f1_score

         train_numerical_path = "../working/3_train_numerical.pkl"

         train_features = pd.read_pickle(train_numerical_path)
         train_labels = pd.read_pickle('../working/1_test_reduced.pkl')

         train_features= np.array(train_features.values).astype(np.float32)
         train_labels = np.array(train_labels.values)[:, 1:].astype(np.float32)

         X_train, X_test, y_train, y_test = train_test_split(train_features, train_labels, test
```

The first technique to be applied is the Random Forest Classifier.

```
In [54]: from sklearn.ensemble import RandomForestClassifier

         clf = RandomForestClassifier(n_estimators=200, n_jobs=-1, verbose=1)
         clf = clf.fit(X_train, y_train)

         y_pred = clf.predict(X_test)
         log_loss(y_test.flatten(), y_pred.flatten(), 1e-15)
         # TS Baseline => 0.0150548
```

```
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.9min
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:  8.6min
[Parallel(n_jobs=-1)]: Done 200 out of 200 | elapsed:  8.9min finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    5.5s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:   24.5s
[Parallel(n_jobs=4)]: Done 200 out of 200 | elapsed:   25.4s finished
```

```
Out[54]: 0.17026122586573778
```

The second one is the Logistic Regression.

```
In [55]: from sklearn.svm import LinearSVC
         from sklearn.multiclass import OneVsRestClassifier

         lr = OneVsRestClassifier(LinearSVC(verbose=1))
         lr = lr.fit(X_train, y_train)

         y_pred = lr.predict(X_test)
         log_loss(y_test.flatten(), y_pred.flatten(), 1e-15)
```

```
[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)

[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)

[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)

[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)

[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)

[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)

[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)

[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)

[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear i
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear i
  "the number of iterations.", ConvergenceWarning)


Out[55]: 1.1428229665120844
```

I have one problem here, the Logistic Regression can't classify all labels the same time, so we have to use the OneVsRest strategy in this algorithm.

The last one is a Multi Layer Perceptron.

```
In [56]: from sklearn.neural_network import MLPClassifier


         mlp = MLPClassifier(hidden_layer_sizes=(128, 256, 64), max_iter=200, alpha=1e-4,
                             verbose=10, tol=1e-4, random_state=1)
         mlp = mlp.fit(X_train, y_train)

         y_pred = mlp.predict(X_test)
         log_loss(y_test.flatten(), y_pred.flatten(), 1e-15)

         # TS Baseline => 0.0150548
```

```
Iteration 1, loss = 7.75827491
Iteration 2, loss = 3.61333115
Iteration 3, loss = 2.86977085
Iteration 4, loss = 2.55908608
Iteration 5, loss = 2.34740342
Iteration 6, loss = 2.21013680
Iteration 7, loss = 2.06775746
Iteration 8, loss = 1.96014643
Iteration 9, loss = 1.86546382
Iteration 10, loss = 1.77282951
Iteration 11, loss = 1.71446989
Iteration 12, loss = 1.66036825
Iteration 13, loss = 1.60122612
Iteration 14, loss = 1.54946367
Iteration 15, loss = 1.51215569
Iteration 16, loss = 1.46565783
Iteration 17, loss = 1.44666310
Iteration 18, loss = 1.40865744
Iteration 19, loss = 1.37638870
Iteration 20, loss = 1.35381125
Iteration 21, loss = 1.34297772
```

```
Iteration 22, loss = 1.32722412
Iteration 23, loss = 1.30942946
Iteration 24, loss = 1.30295837
Iteration 25, loss = 1.28064549
Iteration 26, loss = 1.26658234
Iteration 27, loss = 1.24965484
Iteration 28, loss = 1.23225159
Iteration 29, loss = 1.24332424
Iteration 30, loss = 1.21505399
Iteration 31, loss = 1.20070756
Iteration 32, loss = 1.18989303
Iteration 33, loss = 1.18442198
Iteration 34, loss = 1.17469070
Iteration 35, loss = 1.17972155
Iteration 36, loss = 1.16084436
Iteration 37, loss = 1.14733596
Iteration 38, loss = 1.15210982
Iteration 39, loss = 1.14511686
Iteration 40, loss = 1.12804705
Iteration 41, loss = 1.11930879
Iteration 42, loss = 1.11686127
Iteration 43, loss = 1.10890710
Iteration 44, loss = 1.10738700
Iteration 45, loss = 1.10156827
Iteration 46, loss = 1.09958543
Iteration 47, loss = 1.08393155
Iteration 48, loss = 1.07752340
Iteration 49, loss = 1.07972544
Iteration 50, loss = 1.08430914
Iteration 51, loss = 1.06856237
Iteration 52, loss = 1.06264411
Iteration 53, loss = 1.07310891
Iteration 54, loss = 1.06239809
Iteration 55, loss = 1.05547875
Iteration 56, loss = 1.04909786
Iteration 57, loss = 1.04369439
Iteration 58, loss = 1.04713448
Iteration 59, loss = 1.03706491
Iteration 60, loss = 1.03592832
Iteration 61, loss = 1.03307949
Iteration 62, loss = 1.02680631
Iteration 63, loss = 1.02540112
Iteration 64, loss = 1.02423858
Iteration 65, loss = 1.01607965
Iteration 66, loss = 1.02863344
Iteration 67, loss = 1.00695299
Iteration 68, loss = 1.01546096
Iteration 69, loss = 1.00711120
```

```
Iteration 70, loss = 1.01234037
Training loss did not improve more than tol=0.000100 for two consecutive epochs. Stopping.
```

```
Out[56]: 0.44487915902719338
```

With the original numerical dataset the results are:

Model	Score
RandomForestClassifier	0.17026
LinearSVC	1.14282
MLPClassifier	0.44487

This show the Random Forest can be the best to classify the numeric part of the dataset.

Numerical Part of Dataset scaled with PCA Now with the preprocessed dataset scaled, with PCA on width/height columns.

```
In [63]: train_numerical_path = "../working/9_train_wh_pca.pkl"

         train_features = pd.read_pickle(train_numerical_path)
         train_labels = pd.read_pickle('../working/1_test_reduced.pkl')

         train_features= np.array(train_features.values).astype(np.float32)
         train_labels = np.array(train_labels.values)[:, 1:].astype(np.float32)

         X_train, X_test, y_train, y_test = train_test_split(train_features, train_labels, tes
```

```
In [58]: from sklearn.ensemble import RandomForestClassifier

         clf = RandomForestClassifier(n_estimators=200, n_jobs=-1)
         clf = clf.fit(X_train, y_train)

         y_pred = clf.predict(X_test)
         log_loss(y_test.flatten(), y_pred.flatten(), 1e-15)
         # TS Baseline => 0.0150548
```

```
Out[58]: 0.17170419056527617
```

```
In [64]: from sklearn.svm import LinearSVC
         from sklearn.multiclass import OneVsRestClassifier

         lr = OneVsRestClassifier(LinearSVC(verbose=1))
         lr = lr.fit(X_train, y_train)

         y_pred = lr.predict(X_test)
         log_loss(y_test.flatten(), y_pred.flatten(), 1e-15)
```

```
[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

Out[64]: 0.63848710856789115

In [60]:
```python
from sklearn.neural_network import MLPClassifier


mlp = MLPClassifier(hidden_layer_sizes=(128, 256, 64), max_iter=500, alpha=1e-4,
                    verbose=10, tol=1e-4, random_state=1)
mlp = mlp.fit(X_train, y_train)

y_pred = mlp.predict(X_test)
log_loss(y_test.flatten(), y_pred.flatten(), 1e-15)
```

```
Iteration 1, loss = 1.79037507
Iteration 2, loss = 1.09157372
Iteration 3, loss = 0.96174020
Iteration 4, loss = 0.88272335
Iteration 5, loss = 0.82688660
Iteration 6, loss = 0.78315055
Iteration 7, loss = 0.74824711
Iteration 8, loss = 0.71860139
Iteration 9, loss = 0.69519202
Iteration 10, loss = 0.67120618
Iteration 11, loss = 0.65289768
Iteration 12, loss = 0.63476848
Iteration 13, loss = 0.61863007
Iteration 14, loss = 0.60350771
Iteration 15, loss = 0.59142975
Iteration 16, loss = 0.57941961
Iteration 17, loss = 0.56814110
Iteration 18, loss = 0.55878980
Iteration 19, loss = 0.54792024
Iteration 20, loss = 0.53816135
Iteration 21, loss = 0.52952691
Iteration 22, loss = 0.52001753
Iteration 23, loss = 0.51199664
Iteration 24, loss = 0.50646454
Iteration 25, loss = 0.49892536
Iteration 26, loss = 0.49129905
Iteration 27, loss = 0.48318636
Iteration 28, loss = 0.47883473
Iteration 29, loss = 0.47030448
Iteration 30, loss = 0.46666645
Iteration 31, loss = 0.46135688
Iteration 32, loss = 0.45552384
Iteration 33, loss = 0.45068061
Iteration 34, loss = 0.44459304
Iteration 35, loss = 0.44076101
Iteration 36, loss = 0.43711558
Iteration 37, loss = 0.43005082
Iteration 38, loss = 0.42756694
Iteration 39, loss = 0.42519061
Iteration 40, loss = 0.41961637
Iteration 41, loss = 0.41604565
Iteration 42, loss = 0.41136205
Iteration 43, loss = 0.40672246
Iteration 44, loss = 0.40363534
Iteration 45, loss = 0.40140543
Iteration 46, loss = 0.39721127
Iteration 47, loss = 0.39416864
Iteration 48, loss = 0.39213030
```

```
Iteration 49, loss = 0.38771752
Iteration 50, loss = 0.38515827
Iteration 51, loss = 0.38286685
Iteration 52, loss = 0.37975852
Iteration 53, loss = 0.37342254
Iteration 54, loss = 0.37414174
Iteration 55, loss = 0.37078750
Iteration 56, loss = 0.36660929
Iteration 57, loss = 0.36420223
Iteration 58, loss = 0.36537558
Iteration 59, loss = 0.35922541
Iteration 60, loss = 0.35912881
Iteration 61, loss = 0.35547368
Iteration 62, loss = 0.35417782
Iteration 63, loss = 0.35004362
Iteration 64, loss = 0.34817105
Iteration 65, loss = 0.34914638
Iteration 66, loss = 0.34510494
Iteration 67, loss = 0.34147115
Iteration 68, loss = 0.34165026
Iteration 69, loss = 0.33822035
Iteration 70, loss = 0.33770905
Iteration 71, loss = 0.33368566
Iteration 72, loss = 0.33471202
Iteration 73, loss = 0.32952482
Iteration 74, loss = 0.32753384
Iteration 75, loss = 0.32620791
Iteration 76, loss = 0.32474927
Iteration 77, loss = 0.32319635
Iteration 78, loss = 0.32115629
Iteration 79, loss = 0.31985469
Iteration 80, loss = 0.31960940
Iteration 81, loss = 0.31710124
Iteration 82, loss = 0.31693888
Iteration 83, loss = 0.31674533
Iteration 84, loss = 0.31129953
Iteration 85, loss = 0.31131816
Iteration 86, loss = 0.31182065
Iteration 87, loss = 0.30596217
Iteration 88, loss = 0.30584535
Iteration 89, loss = 0.30505465
Iteration 90, loss = 0.30435697
Iteration 91, loss = 0.30276251
Iteration 92, loss = 0.30005731
Iteration 93, loss = 0.29942497
Iteration 94, loss = 0.30054373
Iteration 95, loss = 0.29545821
Iteration 96, loss = 0.29409279
```

```
Iteration 97, loss = 0.29531742
Iteration 98, loss = 0.29399448
Iteration 99, loss = 0.29293667
Iteration 100, loss = 0.28961353
Iteration 101, loss = 0.28892673
Iteration 102, loss = 0.28879501
Iteration 103, loss = 0.29126610
Iteration 104, loss = 0.28757410
Iteration 105, loss = 0.28534973
Iteration 106, loss = 0.28131946
Iteration 107, loss = 0.28534941
Iteration 108, loss = 0.28291693
Iteration 109, loss = 0.28182879
Training loss did not improve more than tol=0.000100 for two consecutive epochs. Stopping.
```

```
Out[60]: 0.25699470091709481
```

With the scaled numerical dataset the results are:

Model	Score	Original Dataset Score
RandomForestClassifier	0.17170	0.17026
LinearSVC	0.63848	1.14282
MLPClassifier	0.25699	0.44487

The RandomForest stay almost the same but the other two algorithms scored almost the half with the scaling and PCA. The half of the score on the SVC and MLP is kind of expect as they are algorithms that can take benefits from this kind of technique.

Content part of the Dataset Now, with only the content features to see how much the classifiers can do with only them.

```
In [65]: train_only_content_encoding = "../working/4_train_only_content_encoding.pkl"

         train_features = pd.read_pickle(train_only_content_encoding)
         train_features = train_features.astype(np.float32)
         train_features.data = np.nan_to_num(train_features.data)

         train_labels = pd.read_pickle('../working/1_test_reduced.pkl')
         train_labels = np.array(train_labels.values)[:, 1:].astype(np.float32)

         X_train, X_test, y_train, y_test = train_test_split(train_features, train_labels, tes

         del train_features
         del train_labels
         gc.collect()
```

```
Out[65]: 645
```

```
In [66]: from sklearn.ensemble import RandomForestClassifier

         clf = OneVsRestClassifier(RandomForestClassifier(n_estimators=10, n_jobs=-1, verbose=
         clf = clf.fit(X_train, y_train)

         y_pred = clf.predict(X_test)
         log_loss(y_test.flatten(), y_pred.flatten(), 1e-15)
         # TS Baseline => 0.0150548
```

```
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  2.7min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  1.2min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  2.7min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  3.1min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   46.6s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  6.0min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  8.6min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  1.6min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  7.7min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  5.6min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  1.1min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  6.2min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  3.3min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  1.5min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  2.1min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   46.5s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   54.0s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   53.5s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  1.1min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  1.4min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  1.3min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  1.1min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  3.0min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  1.3min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  2.0min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  1.8min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  2.1min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  5.6min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  3.9min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  4.5min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  4.0min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  8.4min finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.7s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.7s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.8s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    1.2s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    3.0s finished
```

```
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    2.6s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    1.5s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    1.1s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.9s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.5s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.8s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.5s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.5s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.5s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    1.6s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    1.0s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    1.1s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    1.0s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    1.8s finished
```

```
Out[66]: 0.23358848574995794
```

As the RandomForestClassifier with all the classes in one Random Forest are taking too long to training and the deadline of the project are coming so I decide to take smaller RandomForest specific for each class with the OneVsRestClassifier.

```
In [67]: from sklearn.svm import LinearSVC
         from sklearn.multiclass import OneVsRestClassifier

         lr = OneVsRestClassifier(LinearSVC(verbose=1))
         lr = lr.fit(X_train, y_train)

         y_pred = lr.predict(X_test)
         log_loss(y_test.flatten(), y_pred.flatten(), 1e-15)
```

```
[LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLi
```

```
Out[67]: 0.19218521850752496
```

```
In [ ]: from sklearn.neural_network import MLPClassifier
```

```
        mlp = MLPClassifier(hidden_layer_sizes=(128, 256, 64), max_iter=500, alpha=1e-4,
                            verbose=10, tol=1e-4, random_state=1)
        mlp = mlp.fit(X_train, y_train)

        y_pred = mlp.predict(X_test)
        log_loss(y_test.flatten(), y_pred.flatten(), 1e-15)
```

```
Iteration 1, loss = 1.58276710
Iteration 2, loss = 0.23860197
Iteration 3, loss = 0.10353965
Iteration 4, loss = 0.07816688
Iteration 5, loss = 0.06693724
Iteration 6, loss = 0.06195221
Iteration 7, loss = 0.05803216
Iteration 8, loss = 0.05802040
Iteration 9, loss = 0.06262843
Iteration 10, loss = 0.06215249
Training loss did not improve more than tol=0.000100 for two consecutive epochs. Stopping.
```

```
Out[ ]: 0.2063457621743853
```

Running the algorithms with only the content part give:

Model	Score	Scaled Dataset Score
RandomForestClassifier	0.23358	0.17170
LinearSVC	0.19218	0.63848
MLPClassifier	0.20634	0.25699

The LinearSVC was the best algorithm to the content part of the problem and was the fastest to training, but the MLPClassifier come close and the hyperparameters can be tunned at the cost of more time.

Complete Scaled with PCA dataset

```
In [2]: x_train_path = "../working/12_train_x.pkl"
        x_test_path = "../working/13_test_x.pkl"
        y_train_path = "../working/14_train_y.pkl"
        y_test_path = "../working/15_test_y.pkl"

        X_train = pd.read_pickle(x_train_path)
        X_test = pd.read_pickle(x_test_path)
        y_train = pd.read_pickle(y_train_path)
        y_test = pd.read_pickle(y_test_path)
```

```
In [ ]: from sklearn.ensemble import RandomForestClassifier
        from sklearn.multiclass import OneVsRestClassifier
```

```
        clf = OneVsRestClassifier(RandomForestClassifier(n_estimators=10, n_jobs=-1, verbose=1)
        clf = clf.fit(X_train, y_train)

        y_pred = clf.predict(X_test)
        log_loss(y_test.flatten(), y_pred.flatten(), 1e-15)
        # TS Baseline => 0.0150548
```

```
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   33.7s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   17.5s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   43.8s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   37.4s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   10.3s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  2.1min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  1.3min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   29.9s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  1.7min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  1.0min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   19.2s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  2.3min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   40.9s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   25.5s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   35.3s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:    9.4s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:    8.9s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   13.7s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   17.9s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   26.1s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   23.6s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   14.9s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   44.1s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   22.7s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   35.7s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   34.8s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   37.4s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  1.1min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   48.0s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   57.8s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   53.7s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  3.6min finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.5s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.5s finished
```

```
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.5s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.5s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.7s finished
```

Out[ ]: 0.23732059011008402

In [ ]:
```
from sklearn.svm import LinearSVC
from sklearn.multiclass import OneVsRestClassifier

lr = OneVsRestClassifier(LinearSVC(verbose=1))
lr = lr.fit(X_train, y_train)

y_pred = lr.predict(X_test)
log_loss(y_test.flatten(), y_pred.flatten(), 1e-15)
```

```
[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear i
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear i
  "the number of iterations.", ConvergenceWarning)


[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

Out[ ]: 0.16703902075213833

In [3]:
```
from sklearn.neural_network import MLPClassifier


mlp = MLPClassifier(hidden_layer_sizes=(128, 256, 64), max_iter=500, alpha=1e-4,
                    verbose=10, tol=1e-4, random_state=1)
mlp = mlp.fit(X_train, y_train)

y_pred = mlp.predict(X_test)
log_loss(y_test.flatten(), y_pred.flatten(), 1e-15)
```

```
Iteration 1, loss = 1.25746951
Iteration 2, loss = 0.24325255
Iteration 3, loss = 0.12384171
Iteration 4, loss = 0.09261436
Iteration 5, loss = 0.07981582
Iteration 6, loss = 0.07286881
Iteration 7, loss = 0.07019086
Iteration 8, loss = 0.06527092
Iteration 9, loss = 0.06604022
Iteration 10, loss = 0.06475756
Iteration 11, loss = 0.06270256
Iteration 12, loss = 0.06387070
Iteration 13, loss = 0.05918358
Iteration 14, loss = 0.06218438
Iteration 15, loss = 0.05994651
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/neural_network/multilayer_perceptron.py:566: Use
  warnings.warn("Training interrupted by user.")
```

```
Out[3]: 0.15581267786636882
```

Running the algorithms with the scaled and content dataset give:

Model	Score	Scaled Dataset Score
RandomForestClassifier	0.23732	0.17170
LinearSVC	0.16703	0.63848
MLPClassifier	0.15581	0.44487

The MLPClassifier was the best algorithm to the content part of the problem but take much longer than the LinearSVC with similar result.

On the implementation stage, I got wrong the metric so I have to rerun all the algorithms besides that sometimes when I ran the classification with the full data give me MemoryError which costs me days and up a VM on a cloud to continue the project plus time to fit the algorithms to date was a limiting factor to experiment more.

1.4.3 Refinement

The usage of standard algorithms with 10% of the dataset to fit in the memory doesn't give me a good score so I have to try different techniques. One is ensemble the best model content only with the best model on numerical features and combines them with a third model. Another one is doing the hash trick and train model against this.

Hash trick all features In this section will do the hash trick on all features.

```
In [9]: from sklearn.feature_extraction import FeatureHasher

        train_bool_transform_path = "../working/1_train_bool_transform.pkl"

        train_features = pd.read_pickle(train_bool_transform_path)
        hasher = FeatureHasher()
        train_features = hasher.transform(train_features.T.to_dict().values())
        train_features = train_features.astype(np.float32)
        train_features.data = np.nan_to_num(train_features.data)

        train_labels = pd.read_pickle('../working/1_test_reduced.pkl')
        train_labels = np.array(train_labels.values)[:, 1:].astype(np.float32)

        X_train, X_test, y_train, y_test = train_test_split(train_features, train_labels, test_

        del train_features
        del train_labels
        gc.collect()
```

```
Out[9]: 7
```

```
In [10]: from sklearn.ensemble import RandomForestClassifier
         from sklearn.multiclass import OneVsRestClassifier

         clf = OneVsRestClassifier(RandomForestClassifier(n_estimators=10, n_jobs=-1, verbose=
         clf = clf.fit(X_train, y_train)

         y_pred = clf.predict(X_test)
         log_loss(y_test.flatten(), y_pred.flatten(), 1e-15)
         # TS Baseline => 0.0150548
```

```
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   35.0s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   15.3s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   42.3s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   36.8s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:    8.6s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  3.2min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  1.7min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   27.7s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  2.4min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  1.1min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   15.4s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  3.4min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   43.7s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   23.3s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   35.8s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:    8.9s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:    7.3s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   11.4s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   15.5s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   26.0s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   21.6s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   11.7s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   50.5s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   18.8s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   31.6s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   32.2s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   32.9s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  1.3min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   57.9s finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  1.3min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  1.2min finished
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:  5.7min finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.5s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.6s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.5s finished
```

```
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.7s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.7s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.5s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.9s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.7s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.7s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.6s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.5s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.6s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.7s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.5s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.5s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.5s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.8s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.6s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.7s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.7s finished
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    1.2s finished
```

```
Out[10]: 0.24398717875384365
```

```
In [11]: from sklearn.svm import LinearSVC
         from sklearn.multiclass import OneVsRestClassifier

         lr = OneVsRestClassifier(LinearSVC(verbose=1))
         lr = lr.fit(X_train, y_train)

         y_pred = lr.predict(X_test)
         log_loss(y_test.flatten(), y_pred.flatten(), 1e-15)
```

```
[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear f
  "the number of iterations.", ConvergenceWarning)


[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)


[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear :
  "the number of iterations.", ConvergenceWarning)
```

```
[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear 
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear 
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear 
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear 
  "the number of iterations.", ConvergenceWarning)


[LibLinear]

/opt/conda/lib/python3.6/site-packages/sklearn/svm/base.py:898: ConvergenceWarning: Liblinear 
  "the number of iterations.", ConvergenceWarning)
```

Out[11]: 3.4017651383839071

In [12]:
```python
from sklearn.neural_network import MLPClassifier


mlp = MLPClassifier(hidden_layer_sizes=(256, 1024, 256), max_iter=500, alpha=1e-4,
                    verbose=10, tol=1e-4, random_state=1)
mlp = mlp.fit(X_train, y_train)

y_pred = mlp.predict(X_test)
log_loss(y_test.flatten(), y_pred.flatten(), 1e-15)
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/neural_network/multilayer_perceptron.py:566: Use
  warnings.warn("Training interrupted by user.")
```

Out[12]: 1.0804636557720153

```
In [13]: from sklearn.linear_model import SGDClassifier
         from sklearn.multiclass import OneVsRestClassifier

         sgd = OneVsRestClassifier(SGDClassifier(loss='log', verbose=1, average=10000))

         sgd = sgd.fit(X_train, y_train)

         y_pred = sgd.predict(X_test)
         log_loss(y_test.flatten(), y_pred.flatten(), 1e-15)
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


-- Epoch 1
Norm: 159476.41, NNZs: 10500, Bias: 0.054533, T: 227800, Avg. loss: 1511727327.260395
Total training time: 0.28 seconds.
-- Epoch 2
Norm: 141376.19, NNZs: 15331, Bias: 0.091196, T: 455600, Avg. loss: 180833220.082084
Total training time: 0.57 seconds.
-- Epoch 3
Norm: 129644.11, NNZs: 19729, Bias: 0.113584, T: 683400, Avg. loss: 106222883.452069
Total training time: 0.86 seconds.
-- Epoch 4
Norm: 124471.49, NNZs: 23704, Bias: 0.135075, T: 911200, Avg. loss: 75660706.553465
Total training time: 1.14 seconds.
-- Epoch 5
Norm: 118959.24, NNZs: 27288, Bias: 0.151102, T: 1139000, Avg. loss: 57826609.425144
Total training time: 1.43 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 24435.66, NNZs: 1656, Bias: -0.066303, T: 227800, Avg. loss: 114464516.484857
Total training time: 0.24 seconds.
-- Epoch 2
Norm: 16026.34, NNZs: 2499, Bias: -0.068957, T: 455600, Avg. loss: 19170375.990464
Total training time: 0.50 seconds.
-- Epoch 3
Norm: 13833.40, NNZs: 3274, Bias: -0.070534, T: 683400, Avg. loss: 11292047.788305
Total training time: 0.76 seconds.
-- Epoch 4
Norm: 17174.82, NNZs: 3944, Bias: -0.069516, T: 911200, Avg. loss: 8178743.017108
Total training time: 1.02 seconds.
-- Epoch 5
Norm: 9377.36, NNZs: 4676, Bias: -0.070222, T: 1139000, Avg. loss: 6296229.704926
```

```
Total training time: 1.30 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 85289.94, NNZs: 23577, Bias: 0.360650, T: 227800, Avg. loss: 4607939006.809593
Total training time: 0.33 seconds.
-- Epoch 2
Norm: 75971.17, NNZs: 36704, Bias: 0.398635, T: 455600, Avg. loss: 604767504.519607
Total training time: 0.62 seconds.
-- Epoch 3
Norm: 72720.90, NNZs: 47834, Bias: 0.411396, T: 683400, Avg. loss: 348544424.691775
Total training time: 0.91 seconds.
-- Epoch 4
Norm: 71049.47, NNZs: 57834, Bias: 0.421419, T: 911200, Avg. loss: 248696157.236344
Total training time: 1.20 seconds.
-- Epoch 5
Norm: 68698.83, NNZs: 66735, Bias: 0.434229, T: 1139000, Avg. loss: 194584812.989235
Total training time: 1.49 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 264276.76, NNZs: 17341, Bias: 0.106115, T: 227800, Avg. loss: 2957572058.970460
Total training time: 0.27 seconds.
-- Epoch 2
Norm: 231310.00, NNZs: 26481, Bias: 0.148421, T: 455600, Avg. loss: 398621365.424369
Total training time: 0.56 seconds.
-- Epoch 3
Norm: 213235.75, NNZs: 34050, Bias: 0.191519, T: 683400, Avg. loss: 232815746.589592
Total training time: 0.84 seconds.
-- Epoch 4
Norm: 201637.99, NNZs: 40776, Bias: 0.222567, T: 911200, Avg. loss: 162678702.276102
Total training time: 1.13 seconds.
-- Epoch 5
Norm: 193042.12, NNZs: 47337, Bias: 0.248727, T: 1139000, Avg. loss: 126902361.730728
Total training time: 1.42 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)
```

```
Norm: 18134.03, NNZs: 459, Bias: -0.268194, T: 227800, Avg. loss: 106009129.381801
Total training time: 0.23 seconds.
-- Epoch 2
Norm: 1297.84, NNZs: 627, Bias: -0.269044, T: 455600, Avg. loss: 2352321.246970
Total training time: 0.47 seconds.
-- Epoch 3
Norm: 8329.10, NNZs: 779, Bias: -0.269562, T: 683400, Avg. loss: 1474097.141012
Total training time: 0.72 seconds.
-- Epoch 4
Norm: 857.93, NNZs: 910, Bias: -0.269663, T: 911200, Avg. loss: 1127163.481593
Total training time: 0.97 seconds.
-- Epoch 5
Norm: 731.04, NNZs: 1069, Bias: -0.270368, T: 1139000, Avg. loss: 750641.467595
Total training time: 1.22 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 669311.90, NNZs: 67122, Bias: 0.853118, T: 227800, Avg. loss: 16241680641.545164
Total training time: 0.29 seconds.
-- Epoch 2
Norm: 577708.64, NNZs: 96261, Bias: 0.876176, T: 455600, Avg. loss: 1965346706.190196
Total training time: 0.58 seconds.
-- Epoch 3
Norm: 521611.82, NNZs: 118216, Bias: 0.888118, T: 683400, Avg. loss: 1150022861.186626
Total training time: 0.87 seconds.
-- Epoch 4
Norm: 481410.18, NNZs: 135631, Bias: 0.898021, T: 911200, Avg. loss: 810041184.288206
Total training time: 1.16 seconds.
-- Epoch 5
Norm: 448781.21, NNZs: 150808, Bias: 0.895011, T: 1139000, Avg. loss: 628255817.287834
Total training time: 1.46 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 286965.49, NNZs: 45817, Bias: 0.573479, T: 227800, Avg. loss: 8253146836.268407
Total training time: 0.28 seconds.
-- Epoch 2
Norm: 222233.14, NNZs: 63805, Bias: 0.763708, T: 455600, Avg. loss: 1069916587.177129
Total training time: 0.58 seconds.
-- Epoch 3
Norm: 205886.41, NNZs: 78838, Bias: 0.886241, T: 683400, Avg. loss: 625776012.543209
```

```
Total training time: 0.87 seconds.
-- Epoch 4
Norm: 188029.32, NNZs: 91541, Bias: 0.982538, T: 911200, Avg. loss: 448556901.436203
Total training time: 1.15 seconds.
-- Epoch 5
Norm: 177495.51, NNZs: 103106, Bias: 1.052451, T: 1139000, Avg. loss: 346696999.815672
Total training time: 1.43 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 47031.87, NNZs: 2154, Bias: -0.206693, T: 227800, Avg. loss: 381880826.442712
Total training time: 0.30 seconds.
-- Epoch 2
Norm: 27830.93, NNZs: 3245, Bias: -0.210218, T: 455600, Avg. loss: 28647765.041517
Total training time: 0.57 seconds.
-- Epoch 3
Norm: 6330.08, NNZs: 4217, Bias: -0.212027, T: 683400, Avg. loss: 15982974.162304
Total training time: 0.84 seconds.
-- Epoch 4
Norm: 5362.12, NNZs: 5243, Bias: -0.214815, T: 911200, Avg. loss: 11435046.786990
Total training time: 1.11 seconds.
-- Epoch 5
Norm: 6103.21, NNZs: 6231, Bias: -0.216864, T: 1139000, Avg. loss: 9017638.416398
Total training time: 1.39 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 186433.90, NNZs: 82598, Bias: -3.040660, T: 227800, Avg. loss: 17470437110.519409
Total training time: 0.29 seconds.
-- Epoch 2
Norm: 163150.63, NNZs: 114428, Bias: -3.527534, T: 455600, Avg. loss: 2066821870.722366
Total training time: 0.57 seconds.
-- Epoch 3
Norm: 153443.19, NNZs: 137705, Bias: -3.809343, T: 683400, Avg. loss: 1214645069.794007
Total training time: 0.86 seconds.
-- Epoch 4
Norm: 144960.46, NNZs: 157500, Bias: -4.026305, T: 911200, Avg. loss: 862592325.548904
Total training time: 1.15 seconds.
-- Epoch 5
Norm: 138721.83, NNZs: 174116, Bias: -4.195735, T: 1139000, Avg. loss: 666585576.522657
Total training time: 1.45 seconds.
```

```
-- Epoch 1

/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)

Norm: 282203.69, NNZs: 24948, Bias: 0.377280, T: 227800, Avg. loss: 4052269731.294174
Total training time: 0.27 seconds.
-- Epoch 2
Norm: 248790.77, NNZs: 35256, Bias: 0.479254, T: 455600, Avg. loss: 495335162.734695
Total training time: 0.55 seconds.
-- Epoch 3
Norm: 232171.64, NNZs: 44037, Bias: 0.554150, T: 683400, Avg. loss: 292517325.051541
Total training time: 0.83 seconds.
-- Epoch 4
Norm: 217037.25, NNZs: 51821, Bias: 0.609869, T: 911200, Avg. loss: 206586217.295473
Total training time: 1.12 seconds.
-- Epoch 5
Norm: 212672.13, NNZs: 59056, Bias: 0.652375, T: 1139000, Avg. loss: 158173567.796391
Total training time: 1.41 seconds.
-- Epoch 1

/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)

Norm: 8306.00, NNZs: 1966, Bias: -0.044695, T: 227800, Avg. loss: 173254287.775998
Total training time: 0.24 seconds.
-- Epoch 2
Norm: 9740.63, NNZs: 2990, Bias: -0.051479, T: 455600, Avg. loss: 22952634.066251
Total training time: 0.49 seconds.
-- Epoch 3
Norm: 7694.11, NNZs: 4057, Bias: -0.056191, T: 683400, Avg. loss: 13330309.510704
Total training time: 0.76 seconds.
-- Epoch 4
Norm: 4394.16, NNZs: 4936, Bias: -0.059395, T: 911200, Avg. loss: 9706450.295365
Total training time: 1.03 seconds.
-- Epoch 5
Norm: 2959.27, NNZs: 5738, Bias: -0.060225, T: 1139000, Avg. loss: 7387179.902297
Total training time: 1.31 seconds.
-- Epoch 1

/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)
```

```
Norm: 571403.05, NNZs: 66234, Bias: 0.980452, T: 227800, Avg. loss: 15244417164.785221
Total training time: 0.29 seconds.
-- Epoch 2
Norm: 502523.52, NNZs: 95303, Bias: 1.019846, T: 455600, Avg. loss: 1947183051.174885
Total training time: 0.58 seconds.
-- Epoch 3
Norm: 448802.15, NNZs: 117027, Bias: 1.031702, T: 683400, Avg. loss: 1142071053.844649
Total training time: 0.88 seconds.
-- Epoch 4
Norm: 412248.06, NNZs: 134856, Bias: 1.043597, T: 911200, Avg. loss: 807393792.007827
Total training time: 1.18 seconds.
-- Epoch 5
Norm: 384898.43, NNZs: 149982, Bias: 1.046016, T: 1139000, Avg. loss: 627321547.524737
Total training time: 1.47 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 285819.63, NNZs: 19482, Bias: -0.246042, T: 227800, Avg. loss: 3267202247.344995
Total training time: 0.27 seconds.
-- Epoch 2
Norm: 252287.78, NNZs: 28713, Bias: -0.192541, T: 455600, Avg. loss: 410823481.142027
Total training time: 0.55 seconds.
-- Epoch 3
Norm: 233824.51, NNZs: 36519, Bias: -0.148763, T: 683400, Avg. loss: 241490684.983383
Total training time: 0.84 seconds.
-- Epoch 4
Norm: 221586.53, NNZs: 43399, Bias: -0.117422, T: 911200, Avg. loss: 170702893.962658
Total training time: 1.13 seconds.
-- Epoch 5
Norm: 213824.66, NNZs: 49618, Bias: -0.094043, T: 1139000, Avg. loss: 132025236.672636
Total training time: 1.42 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 48722.53, NNZs: 4289, Bias: 0.077193, T: 227800, Avg. loss: 526008615.594018
Total training time: 0.25 seconds.
-- Epoch 2
Norm: 53302.61, NNZs: 6494, Bias: 0.090873, T: 455600, Avg. loss: 71455462.960552
Total training time: 0.52 seconds.
-- Epoch 3
Norm: 45865.96, NNZs: 8632, Bias: 0.096301, T: 683400, Avg. loss: 41596670.321194
```

```
Total training time: 0.78 seconds.
-- Epoch 4
Norm: 44741.29, NNZs: 10644, Bias: 0.100328, T: 911200, Avg. loss: 29127700.414282
Total training time: 1.06 seconds.
-- Epoch 5
Norm: 42951.50, NNZs: 12568, Bias: 0.104483, T: 1139000, Avg. loss: 22692337.568333
Total training time: 1.34 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 154614.43, NNZs: 12888, Bias: -0.236115, T: 227800, Avg. loss: 2548533842.448460
Total training time: 0.27 seconds.
-- Epoch 2
Norm: 128510.43, NNZs: 20618, Bias: -0.184183, T: 455600, Avg. loss: 327534339.229750
Total training time: 0.55 seconds.
-- Epoch 3
Norm: 124743.65, NNZs: 27438, Bias: -0.148146, T: 683400, Avg. loss: 189161543.589938
Total training time: 0.84 seconds.
-- Epoch 4
Norm: 115152.44, NNZs: 33497, Bias: -0.114768, T: 911200, Avg. loss: 135991076.714467
Total training time: 1.13 seconds.
-- Epoch 5
Norm: 107300.78, NNZs: 39116, Bias: -0.090520, T: 1139000, Avg. loss: 105454153.297097
Total training time: 1.41 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 5766.78, NNZs: 481, Bias: -0.042247, T: 227800, Avg. loss: 12614107.602296
Total training time: 0.23 seconds.
-- Epoch 2
Norm: 36273.19, NNZs: 638, Bias: -0.041847, T: 455600, Avg. loss: 2791610.216660
Total training time: 0.47 seconds.
-- Epoch 3
Norm: 5355.19, NNZs: 770, Bias: -0.041687, T: 683400, Avg. loss: 1652501.466617
Total training time: 0.70 seconds.
-- Epoch 4
Norm: 3931.27, NNZs: 936, Bias: -0.041878, T: 911200, Avg. loss: 1120344.967239
Total training time: 0.95 seconds.
-- Epoch 5
Norm: 5111.27, NNZs: 1071, Bias: -0.042013, T: 1139000, Avg. loss: 806141.331374
Total training time: 1.20 seconds.
```

```
-- Epoch 1

/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)

Norm: 8396.59, NNZs: 475, Bias: -0.045700, T: 227800, Avg. loss: 9099431.205571
Total training time: 0.23 seconds.
-- Epoch 2
Norm: 29166.44, NNZs: 662, Bias: -0.043718, T: 455600, Avg. loss: 2930155.145271
Total training time: 0.47 seconds.
-- Epoch 3
Norm: 9168.67, NNZs: 809, Bias: -0.042586, T: 683400, Avg. loss: 1675543.499822
Total training time: 0.70 seconds.
-- Epoch 4
Norm: 10561.42, NNZs: 999, Bias: -0.042605, T: 911200, Avg. loss: 1183246.783442
Total training time: 0.95 seconds.
-- Epoch 5
Norm: 10243.99, NNZs: 1138, Bias: -0.041897, T: 1139000, Avg. loss: 943009.627979
Total training time: 1.21 seconds.
-- Epoch 1

/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)

Norm: 22316.19, NNZs: 1517, Bias: -0.051861, T: 227800, Avg. loss: 211781458.423686
Total training time: 0.23 seconds.
-- Epoch 2
Norm: 21266.62, NNZs: 2362, Bias: -0.048894, T: 455600, Avg. loss: 21496094.960950
Total training time: 0.48 seconds.
-- Epoch 3
Norm: 27618.83, NNZs: 3280, Bias: -0.050758, T: 683400, Avg. loss: 12083385.668201
Total training time: 0.74 seconds.
-- Epoch 4
Norm: 19403.97, NNZs: 4093, Bias: -0.050472, T: 911200, Avg. loss: 8996503.939986
Total training time: 1.01 seconds.
-- Epoch 5
Norm: 18516.66, NNZs: 4813, Bias: -0.049151, T: 1139000, Avg. loss: 6898769.049111
Total training time: 1.27 seconds.
-- Epoch 1

/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)
```

```
Norm: 73526.27, NNZs: 4067, Bias: 0.080730, T: 227800, Avg. loss: 677514362.794837
Total training time: 0.25 seconds.
-- Epoch 2
Norm: 57973.83, NNZs: 6894, Bias: 0.090335, T: 455600, Avg. loss: 88887524.455368
Total training time: 0.51 seconds.
-- Epoch 3
Norm: 54160.18, NNZs: 9348, Bias: 0.097010, T: 683400, Avg. loss: 52187710.164640
Total training time: 0.77 seconds.
-- Epoch 4
Norm: 52046.58, NNZs: 11712, Bias: 0.102607, T: 911200, Avg. loss: 37562569.607137
Total training time: 1.05 seconds.
-- Epoch 5
Norm: 51338.74, NNZs: 13958, Bias: 0.105707, T: 1139000, Avg. loss: 28827662.686144
Total training time: 1.33 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 156196.50, NNZs: 8563, Bias: -0.051427, T: 227800, Avg. loss: 1600864720.029404
Total training time: 0.27 seconds.
-- Epoch 2
Norm: 126428.01, NNZs: 14367, Bias: -0.019916, T: 455600, Avg. loss: 217099905.525331
Total training time: 0.55 seconds.
-- Epoch 3
Norm: 117416.12, NNZs: 19318, Bias: 0.000454, T: 683400, Avg. loss: 128546695.181213
Total training time: 0.84 seconds.
-- Epoch 4
Norm: 109267.51, NNZs: 23971, Bias: 0.017843, T: 911200, Avg. loss: 91891878.814558
Total training time: 1.12 seconds.
-- Epoch 5
Norm: 102788.84, NNZs: 28025, Bias: 0.036748, T: 1139000, Avg. loss: 71752093.076750
Total training time: 1.41 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 105427.53, NNZs: 7355, Bias: -0.082396, T: 227800, Avg. loss: 1450366901.780401
Total training time: 0.26 seconds.
-- Epoch 2
Norm: 97915.94, NNZs: 12459, Bias: -0.058458, T: 455600, Avg. loss: 181124047.885755
Total training time: 0.54 seconds.
-- Epoch 3
Norm: 95531.46, NNZs: 16945, Bias: -0.054149, T: 683400, Avg. loss: 104038074.744420
```

```
Total training time: 0.81 seconds.
-- Epoch 4
Norm: 93127.95, NNZs: 21173, Bias: -0.045034, T: 911200, Avg. loss: 75018454.963161
Total training time: 1.10 seconds.
-- Epoch 5
Norm: 90463.72, NNZs: 25150, Bias: -0.038306, T: 1139000, Avg. loss: 57830495.704056
Total training time: 1.38 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 38932.94, NNZs: 1665, Bias: -0.044357, T: 227800, Avg. loss: 179121289.833506
Total training time: 0.24 seconds.
-- Epoch 2
Norm: 40139.01, NNZs: 2830, Bias: -0.046077, T: 455600, Avg. loss: 28036432.452848
Total training time: 0.49 seconds.
-- Epoch 3
Norm: 34623.33, NNZs: 3764, Bias: -0.043317, T: 683400, Avg. loss: 16407792.933124
Total training time: 0.75 seconds.
-- Epoch 4
Norm: 34897.42, NNZs: 4726, Bias: -0.041498, T: 911200, Avg. loss: 11666458.098299
Total training time: 1.02 seconds.
-- Epoch 5
Norm: 34237.32, NNZs: 5670, Bias: -0.040909, T: 1139000, Avg. loss: 9065547.933293
Total training time: 1.28 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 131682.77, NNZs: 18940, Bias: 0.257348, T: 227800, Avg. loss: 3930645644.485791
Total training time: 0.29 seconds.
-- Epoch 2
Norm: 125287.17, NNZs: 30203, Bias: 0.290550, T: 455600, Avg. loss: 492066351.761178
Total training time: 0.58 seconds.
-- Epoch 3
Norm: 119573.01, NNZs: 39586, Bias: 0.329368, T: 683400, Avg. loss: 291579243.868587
Total training time: 0.86 seconds.
-- Epoch 4
Norm: 115098.17, NNZs: 48318, Bias: 0.345073, T: 911200, Avg. loss: 205661737.803822
Total training time: 1.15 seconds.
-- Epoch 5
Norm: 110251.58, NNZs: 56504, Bias: 0.361655, T: 1139000, Avg. loss: 160532136.451204
Total training time: 1.44 seconds.
```

```
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 60041.35, NNZs: 3797, Bias: -0.006548, T: 227800, Avg. loss: 496295724.014394
Total training time: 0.25 seconds.
-- Epoch 2
Norm: 52302.59, NNZs: 6254, Bias: -0.002167, T: 455600, Avg. loss: 74011273.606121
Total training time: 0.52 seconds.
-- Epoch 3
Norm: 52948.78, NNZs: 8470, Bias: 0.002115, T: 683400, Avg. loss: 43236956.434871
Total training time: 0.80 seconds.
-- Epoch 4
Norm: 50384.86, NNZs: 10473, Bias: 0.005158, T: 911200, Avg. loss: 30537639.472057
Total training time: 1.08 seconds.
-- Epoch 5
Norm: 51832.75, NNZs: 12383, Bias: 0.006731, T: 1139000, Avg. loss: 23624056.060721
Total training time: 1.36 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 78893.69, NNZs: 12857, Bias: -0.266314, T: 227800, Avg. loss: 2427796448.369153
Total training time: 0.27 seconds.
-- Epoch 2
Norm: 69435.28, NNZs: 21203, Bias: -0.260723, T: 455600, Avg. loss: 329674962.228450
Total training time: 0.56 seconds.
-- Epoch 3
Norm: 60342.09, NNZs: 28257, Bias: -0.241055, T: 683400, Avg. loss: 195517176.371807
Total training time: 0.85 seconds.
-- Epoch 4
Norm: 56156.68, NNZs: 34789, Bias: -0.232878, T: 911200, Avg. loss: 138002866.605738
Total training time: 1.13 seconds.
-- Epoch 5
Norm: 54548.37, NNZs: 40929, Bias: -0.226618, T: 1139000, Avg. loss: 107216380.948560
Total training time: 1.42 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)
```

```
Norm: 39250.04, NNZs: 11513, Bias: 0.041498, T: 227800, Avg. loss: 1696825581.016360
Total training time: 0.27 seconds.
-- Epoch 2
Norm: 34554.54, NNZs: 19029, Bias: 0.000173, T: 455600, Avg. loss: 257209688.357834
Total training time: 0.55 seconds.
-- Epoch 3
Norm: 33953.47, NNZs: 25364, Bias: -0.013503, T: 683400, Avg. loss: 149878534.622579
Total training time: 0.83 seconds.
-- Epoch 4
Norm: 33455.27, NNZs: 31323, Bias: -0.025419, T: 911200, Avg. loss: 106006626.688067
Total training time: 1.12 seconds.
-- Epoch 5
Norm: 28098.39, NNZs: 36744, Bias: -0.037922, T: 1139000, Avg. loss: 83457874.418757
Total training time: 1.41 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 86268.99, NNZs: 11888, Bias: -0.096420, T: 227800, Avg. loss: 2088981715.618895
Total training time: 0.27 seconds.
-- Epoch 2
Norm: 77519.88, NNZs: 19520, Bias: -0.105745, T: 455600, Avg. loss: 285990551.314212
Total training time: 0.56 seconds.
-- Epoch 3
Norm: 77189.51, NNZs: 26277, Bias: -0.114848, T: 683400, Avg. loss: 166977351.819747
Total training time: 0.85 seconds.
-- Epoch 4
Norm: 70099.71, NNZs: 32390, Bias: -0.128862, T: 911200, Avg. loss: 118359386.699660
Total training time: 1.13 seconds.
-- Epoch 5
Norm: 67131.07, NNZs: 37893, Bias: -0.136334, T: 1139000, Avg. loss: 91762665.829746
Total training time: 1.42 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 52163.15, NNZs: 43485, Bias: -1.992134, T: 227800, Avg. loss: 7029045302.288929
Total training time: 0.29 seconds.
-- Epoch 2
Norm: 44312.88, NNZs: 62349, Bias: -2.393547, T: 455600, Avg. loss: 897604845.864154
Total training time: 0.58 seconds.
-- Epoch 3
Norm: 44628.26, NNZs: 77838, Bias: -2.625579, T: 683400, Avg. loss: 525473931.966417
```

```
Total training time: 0.87 seconds.
-- Epoch 4
Norm: 38313.19, NNZs: 91183, Bias: -2.798667, T: 911200, Avg. loss: 375899769.555714
Total training time: 1.16 seconds.
-- Epoch 5
Norm: 34219.11, NNZs: 103234, Bias: -2.931793, T: 1139000, Avg. loss: 290887712.284067
Total training time: 1.45 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 477334.09, NNZs: 27920, Bias: 0.029255, T: 227800, Avg. loss: 5126047324.886663
Total training time: 0.28 seconds.
-- Epoch 2
Norm: 447301.50, NNZs: 41277, Bias: 0.150832, T: 455600, Avg. loss: 665449076.672161
Total training time: 0.57 seconds.
-- Epoch 3
Norm: 427081.51, NNZs: 52351, Bias: 0.217469, T: 683400, Avg. loss: 380273913.019508
Total training time: 0.86 seconds.
-- Epoch 4
Norm: 407363.36, NNZs: 62114, Bias: 0.263071, T: 911200, Avg. loss: 268664184.950236
Total training time: 1.15 seconds.
-- Epoch 5
Norm: 391118.61, NNZs: 70990, Bias: 0.285474, T: 1139000, Avg. loss: 207952457.129444
Total training time: 1.44 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 414025.86, NNZs: 33939, Bias: 0.714421, T: 227800, Avg. loss: 6393898432.696514
Total training time: 0.28 seconds.
-- Epoch 2
Norm: 380531.08, NNZs: 49445, Bias: 0.857887, T: 455600, Avg. loss: 831389434.290046
Total training time: 0.57 seconds.
-- Epoch 3
Norm: 358468.22, NNZs: 62163, Bias: 0.946328, T: 683400, Avg. loss: 483914642.749418
Total training time: 0.86 seconds.
-- Epoch 4
Norm: 339711.82, NNZs: 73718, Bias: 0.995048, T: 911200, Avg. loss: 341175943.434772
Total training time: 1.15 seconds.
-- Epoch 5
Norm: 321527.46, NNZs: 83863, Bias: 1.027415, T: 1139000, Avg. loss: 265733028.558601
Total training time: 1.44 seconds.
```

```
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 599781.31, NNZs: 56180, Bias: -4.242003, T: 227800, Avg. loss: 12317232009.766893
Total training time: 0.29 seconds.
-- Epoch 2
Norm: 500557.82, NNZs: 82572, Bias: -4.835158, T: 455600, Avg. loss: 1442742583.405521
Total training time: 0.59 seconds.
-- Epoch 3
Norm: 438513.07, NNZs: 102480, Bias: -5.168323, T: 683400, Avg. loss: 845192376.623882
Total training time: 0.88 seconds.
-- Epoch 4
Norm: 396728.89, NNZs: 119089, Bias: -5.405359, T: 911200, Avg. loss: 591398942.960882
Total training time: 1.17 seconds.
-- Epoch 5
Norm: 365872.91, NNZs: 133526, Bias: -5.586415, T: 1139000, Avg. loss: 459207686.132596
Total training time: 1.47 seconds.
-- Epoch 1


/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureW
  "and default tol will be 1e-3." % type(self), FutureWarning)


Norm: 1141795.13, NNZs: 195844, Bias: -4.757379, T: 227800, Avg. loss: 56016708728.726463
Total training time: 0.30 seconds.
-- Epoch 2
Norm: 970167.04, NNZs: 255851, Bias: -5.336100, T: 455600, Avg. loss: 7021412580.668544
Total training time: 0.60 seconds.
-- Epoch 3
Norm: 856056.66, NNZs: 283118, Bias: -5.664931, T: 683400, Avg. loss: 4078360629.115651
Total training time: 0.89 seconds.
-- Epoch 4
Norm: 768550.04, NNZs: 297251, Bias: -5.871928, T: 911200, Avg. loss: 2884521537.502388
Total training time: 1.19 seconds.
-- Epoch 5
Norm: 696561.66, NNZs: 305007, Bias: -6.004474, T: 1139000, Avg. loss: 2235504587.709351
Total training time: 1.49 seconds.


Out[13]: 1.0488040514554187
```

Running the algorithms with the hashing trick give the following scores:

Model	Score	Scaled + Content Dataset Score
RandomForestClassifier	0.24398	0.23732
LinearSVC	3.40176	0.16703
MLPClassifier	1.08046	0.15581
SGDClassifier	1.04880	-

As we see, no good for anyone of the algorithms with the hash trick. I will continue to investigate to choose the best to feed the full data. The next one is the ensemble technique.

Ensemble the bests models I will choose the Random Forest to the numerical part of the dataset and LinearSVC to the content part of the dataset using the prediction of both as input to another Random Forest that will make the prediction.

```
In [14]: X_meta = []
         X_meta_test = []

In [15]: # loading the numerical dataset

         train_numerical_path = "../working/9_train_wh_pca.pkl"

         train_features = pd.read_pickle(train_numerical_path)
         train_labels = pd.read_pickle('../working/1_test_reduced.pkl')

         train_features= np.array(train_features.values).astype(np.float32)
         train_labels = np.array(train_labels.values)[:, 1:].astype(np.float32)

         X_train, X_test, y_train, y_test = train_test_split(train_features, train_labels, tes

         del train_features
         del train_labels
         gc.collect()

Out[15]: 24

In [17]: # training the numerical meta

         from sklearn.ensemble import RandomForestClassifier

         rf = OneVsRestClassifier(RandomForestClassifier(n_estimators=10, verbose=1))
         rf = rf.fit(X_train, y_train)

         X_meta.append(rf.predict_proba(X_train))
         X_meta_test.append(rf.predict_proba(X_test))

[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    8.4s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    6.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    6.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    7.3s finished
```

```
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    4.7s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    9.8s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:   11.3s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    5.9s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    9.6s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:   12.3s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    6.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:   10.0s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    9.9s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    5.8s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    8.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    4.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    2.9s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    4.9s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    5.4s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    7.5s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    6.9s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    4.9s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    8.0s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    7.6s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    8.9s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    8.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:   10.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    9.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    8.9s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    8.6s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    9.9s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:   16.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.4s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
```

```
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.6s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.3s finished
```

```
In [18]: # loading the content dataset
```

```
        train_only_content_encoding = "../working/4_train_only_content_encoding.pkl"

        train_features = pd.read_pickle(train_only_content_encoding)
        train_features = train_features.astype(np.float32)
        train_features.data = np.nan_to_num(train_features.data)

        train_labels = pd.read_pickle('../working/1_test_reduced.pkl')
        train_labels = np.array(train_labels.values)[:, 1:].astype(np.float32)

        X_train, X_test, y_train, y_test = train_test_split(train_features, train_labels, test

        del train_features
        del train_labels
        gc.collect()
```

Out[18]: 395099

```
In [19]: # training the content meta

         from sklearn.svm import LinearSVC
         from sklearn.multiclass import OneVsRestClassifier

         svc = OneVsRestClassifier(LinearSVC(verbose=1))
         svc = svc.fit(X_train, y_train)

         X_meta.append(svc.decision_function(X_train))
         X_meta_test.append(svc.decision_function(X_test))
```

```
[LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLi
```

```
In [20]: X_meta[0].shape
```

Out[20]: (227800, 32)

```
In [21]: X_meta[1].shape
```

Out[21]: (227800, 32)

```
In [22]: # bring together both meta
         X_all_meta = np.column_stack(X_meta)
```

```
In [23]: X_all_meta.shape
```

Out[23]: (227800, 64)

```
In [24]: X_all_meta_test = np.column_stack(X_meta_test)
         X_all_meta_test.shape
```

Out[24]: (112200, 64)

```
In [25]: # trainig the new classifier

         meta = OneVsRestClassifier(RandomForestClassifier(n_estimators=30, verbose=1))
         meta = meta.fit(X_all_meta, y_train)

         y_pred = meta.predict(X_all_meta_test)
         log_loss(y_test.flatten(), y_pred.flatten(), 1e-15)
```

```
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   11.7s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   13.6s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   14.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   14.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    9.9s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   21.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   20.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   18.8s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   26.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   21.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   11.6s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   22.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   15.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   12.8s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   16.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    9.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    8.6s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   11.8s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    9.9s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   13.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   12.6s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   11.0s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   13.8s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   11.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   13.4s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   12.7s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   14.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   15.0s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   16.7s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   16.7s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   18.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   45.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
```

```
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.4s finished
```

Out[25]: 0.1499251635084386

In [54]:
```
# cross validating this classifier

from sklearn.metrics import log_loss, make_scorer
from sklearn.cross_validation import cross_val_score

log_loss_scorer = make_scorer(log_loss, needs_proba = True)
rf_cv = OneVsRestClassifier(RandomForestClassifier(n_estimators=30, verbose=1, n_jobs=

scores = cross_val_score(rf_cv, X_all_meta, y_train, cv = 4, n_jobs = 1, scoring = log
```

```
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   11.8s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   10.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   12.9s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   12.9s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    7.8s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   20.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   32.6s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   21.6s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   24.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   28.0s finished
```

```
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   11.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   21.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   14.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   10.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   12.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    9.7s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    7.7s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   10.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   11.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   12.6s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   13.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   12.9s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   23.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   15.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   18.0s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   22.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   14.8s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   15.6s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   15.8s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   14.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   15.4s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   41.7s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
```

```
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   10.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   12.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   14.7s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   15.8s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   10.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   21.0s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   21.9s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   20.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   24.9s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   23.7s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   24.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   41.6s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   27.6s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   19.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   24.4s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   15.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   15.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   18.7s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   19.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   19.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   21.0s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   16.8s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   24.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   15.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   13.0s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   10.9s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   14.8s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   29.6s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   31.9s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   34.6s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   30.0s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:  1.0min finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.4s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.4s finished
```

```
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   21.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   15.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   16.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   14.6s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    9.6s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   19.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   19.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   17.0s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   24.9s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   28.6s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   23.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   47.4s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   22.0s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   18.8s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   21.8s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   14.6s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   12.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   10.8s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   10.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   12.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   11.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   17.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   25.8s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   19.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   23.7s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   23.8s finished
```

```
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   25.8s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   32.4s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   26.4s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   32.4s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   16.9s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:  1.1min finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   19.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   14.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   12.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   13.4s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    8.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   22.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   20.9s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   14.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   21.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   19.4s finished
```

```
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    9.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   18.9s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   12.9s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   11.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    9.9s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    7.8s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    8.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   11.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    9.8s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   11.7s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   10.4s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    9.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   13.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   10.4s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   11.3s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   11.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   12.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   13.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   10.6s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   14.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   15.5s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:   36.4s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
```

```
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.2s finished
```

```
In [84]: print(np.mean(scores))
         print(np.std(scores))
```

```
0.20038512069
0.00177486788901
```

This ensemble classifier is the best I can get with 10% of the dataset, so I will train him with 100% of the dataset and then try to improve with hyperparameter tuning.

First, preprocessing all the dataset, them training the ensemble model.

```
In [28]: train_features = pre.transform_all()
```

```
In [29]: train_labels = d.read_train_labels()
         # drop y14, it is always 0
         train_labels.drop(labels=['y14'], axis="columns", inplace=True)
         train_labels = np.array(train_labels.values)[:, 1:].astype(np.float32)

         X_train, X_test, X_content_train, X_content_test, y_train, y_test = \
             train_test_split(train_features, content_df, train_labels, test_size=0.25, random_

         del train_labels
         del train_features
         del content_df
         del pca_wh
         del float_scaler
         del int_scaler
         gc.collect()
```

```
Out[29]: 29
```

```
In [48]: pre.save_all(X_train, X_test, X_content_train, X_content_test, y_train, y_test)
```

```
In [30]: X_train, X_test, X_content_train, X_content_test, y_train, y_test = pre.load_all()
```

```
In [31]: print('X_train.shape={}'.format(X_train.shape))
         print('X_test.shape={}'.format(X_test.shape))
         print('X_content_train.shape={}'.format(X_content_train.shape))
         print('X_content_test.shape={}'.format(X_content_test.shape))
         print('y_train.shape={}'.format(y_train.shape))
         print('y_test.shape={}'.format(y_test.shape))
```

```
X_train.shape=(1275000, 130)
X_test.shape=(425000, 130)
X_content_train.shape=(1275000, 1595351)
X_content_test.shape=(425000, 1595351)
y_train.shape=(1275000, 32)
y_test.shape=(425000, 32)
```

Starting the training

```
In [32]: X_meta = []
         X_meta_test = []
```

```
In [33]: # training the numerical meta
         from sklearn.multiclass import OneVsRestClassifier
         from sklearn.ensemble import RandomForestClassifier

         rf = OneVsRestClassifier(RandomForestClassifier(n_estimators=100, verbose=1, n_jobs=-1
         rf = rf.fit(X_train, y_train)

         X_meta.append(rf.predict_proba(X_train))
         X_meta_test.append(rf.predict_proba(X_test))
```

```
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.6min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  3.6min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  3.3min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  2.9min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  2.6min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  2.5min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.8min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  4.2min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  2.2min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  5.0min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  3.1min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.9min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  4.5min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  2.3min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  5.5min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  2.8min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.8min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  4.1min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.9min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  4.4min finished
```

```
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:   55.4s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  2.2min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.6min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  3.7min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:   55.0s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  2.1min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:   50.9s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  2.1min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:   59.1s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  2.3min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  2.6min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  3.2min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  3.4min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:   58.7s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  2.3min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  3.6min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.6min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  3.6min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  4.0min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  3.5min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.9min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  4.2min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  4.2min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  4.1min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  3.6min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.9min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  4.3min finished
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  3.0min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  6.9min finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    3.2s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    7.4s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    2.3s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    5.5s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    2.8s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    6.3s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    2.0s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    5.1s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.7s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    4.0s finished
```

```
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    4.8s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:   11.3s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    5.2s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:   12.1s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    2.4s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    5.7s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    5.1s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:   12.0s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    4.7s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:   11.2s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    2.0s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    4.7s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    5.0s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:   11.5s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    3.6s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    8.4s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.9s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    4.3s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    3.1s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    7.4s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.8s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    4.0s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.8s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    4.1s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.9s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    4.5s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    2.0s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    4.7s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    2.7s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    6.2s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    2.9s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    6.5s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.7s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    3.9s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    3.5s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    8.1s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    2.4s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    5.5s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    3.3s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    7.8s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    3.1s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    7.0s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    3.4s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    7.9s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    4.5s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:   10.5s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    3.8s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    9.0s finished
```

```
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    3.9s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    9.1s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    4.3s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:   10.1s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:   11.0s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:   25.7s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.1s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    2.5s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.8s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    1.8s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.9s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    2.1s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.7s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    1.7s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.6s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    1.3s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.6s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    3.8s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.7s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    4.0s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.8s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    1.9s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.7s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    4.0s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.6s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    3.6s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.7s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    1.6s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.7s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    3.8s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.2s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    2.7s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.6s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    1.4s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.1s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    2.5s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.6s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    1.3s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.6s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    1.4s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.6s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    1.5s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.7s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    1.6s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.9s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    2.1s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.9s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    2.2s finished
```

```
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.6s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    1.3s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.1s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    2.7s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.8s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    1.8s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.0s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    2.5s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.0s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    2.3s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.1s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    2.6s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.5s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    3.5s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.2s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    3.0s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.3s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    3.0s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    1.5s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    3.5s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    3.8s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    8.7s finished
```

```
In [34]: from sklearn.svm import LinearSVC
         from sklearn.multiclass import OneVsRestClassifier

         svc = OneVsRestClassifier(LinearSVC(verbose=1))
         svc = svc.fit(X_content_train, y_train)

         X_meta.append(svc.decision_function(X_content_train))
         X_meta_test.append(svc.decision_function(X_content_test))
```

```
[LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLi
```

```
In [35]: X_all_meta = np.column_stack(X_meta)
         X_all_meta_test = np.column_stack(X_meta_test)
```

```
In [36]: # trainig the new classifier

         meta_clf = OneVsRestClassifier(RandomForestClassifier(n_estimators=30, verbose=1, n_j
         meta_clf = meta_clf.fit(X_all_meta, y_train)

         y_pred = meta_clf.predict(X_all_meta_test)
         log_loss(y_test.flatten(), y_pred.flatten(), 1e-15)
```

```
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   45.9s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   45.9s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   33.6s finished
```

```
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   43.1s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   31.5s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:  1.1min finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:  1.1min finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   53.9s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:  1.4min finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:  1.2min finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   29.2s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:  1.3min finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   56.2s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   40.8s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   31.5s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   28.2s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   37.3s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   33.5s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   27.0s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   30.1s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   30.4s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   32.0s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   33.6s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   33.1s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   39.6s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   29.5s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   37.3s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   46.8s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   45.7s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   57.1s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   53.0s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:  2.5min finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.5s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.5s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.5s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.5s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
```

```
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.4s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.7s finished
```

Out[36]: 0.090898680030117099

```
In [37]: pickle.dump(rf, open('../working/25_base_rf.pkl', 'wb'))
         pickle.dump(svc, open('../working/26_base_svc.pkl', 'wb'))
         pickle.dump(meta, open('../working/27_meta_rf.pkl', 'wb'))
```

Out[37]: ['../working/19_meta_rf.pkl']

1.5 IV. Results

1.5.1 Model Evaluation and Validation

```
In [51]: # cross validating this classifier

         from sklearn.metrics import log_loss, make_scorer
         from sklearn.cross_validation import cross_val_score

         log_loss_scorer = make_scorer(log_loss, needs_proba = True)
         rf_cv = OneVsRestClassifier(RandomForestClassifier(n_estimators=30, verbose=1, n_jobs=

         scores = cross_val_score(rf_cv, X_all_meta, y_train, cv = 4, n_jobs = 1, scoring = log
         print(np.mean(scores))
         print(np.std(scores))
```

```
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   31.7s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   30.0s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   21.1s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   30.6s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   24.5s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   55.2s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   46.7s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   41.2s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:  1.0min finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   56.9s finished
```

```
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   23.7s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   57.5s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   44.7s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   24.5s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   26.1s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   19.7s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   22.2s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   22.2s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   26.7s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   25.7s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   23.6s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   25.6s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   24.2s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   27.1s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   29.5s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   27.7s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   29.3s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   32.9s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   34.2s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   39.2s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   38.5s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:  1.9min finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
```

```
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.4s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   34.7s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   34.4s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   30.7s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   32.8s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   22.3s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   55.0s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   50.5s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   37.2s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:  1.1min finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:  1.0min finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   21.7s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   58.0s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   40.5s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   26.7s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   26.1s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   19.9s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   24.3s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   18.9s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   23.5s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   29.5s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   25.6s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   19.7s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   28.3s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   21.7s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   26.5s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   26.4s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   26.2s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   32.4s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   34.5s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   41.6s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   38.6s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:  1.8min finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
```

```
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.4s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   36.0s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   31.4s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   18.1s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   35.4s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   24.5s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   54.6s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   47.2s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   41.6s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:  1.2min finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   57.8s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   24.5s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   54.8s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   39.8s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   30.6s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   27.5s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   26.9s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   22.6s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   23.2s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   20.1s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   25.0s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   22.6s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   29.3s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   29.3s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   24.6s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   25.6s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   28.8s finished
```

```
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   25.6s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   33.2s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   32.3s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   42.4s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   39.6s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:  1.9min finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.4s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   32.8s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   30.6s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   28.4s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   34.7s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   27.3s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   53.4s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   46.3s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   39.8s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:  1.0min finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   51.4s finished
```

```
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   23.0s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:  1.0min finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   39.9s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   25.9s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   30.9s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   21.2s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   21.9s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   19.1s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   24.5s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   25.1s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   24.9s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   23.2s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   22.4s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   21.5s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   25.2s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   28.1s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   23.2s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   31.2s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   36.1s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   39.5s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   38.8s finished
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:  1.9min finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
```

```
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.3s finished
[Parallel(n_jobs=4)]: Done  30 out of  30 | elapsed:    0.4s finished


0.202689890375
0.000314707835958
```

The final ensenble model pass to a evaluation of the score by cross-validation and the score has a 0.20268 as median score and a 0.00031 as standard deviation show that is a robust model and can be trusted.

1.5.2 Justification

My final solution beat the Random Benchmark and the All Halves Benchmark but can't beat the TS Baseline Benchmark, my best model scores were 0.09285 and the TS Baseline was 0.0150548. My results are not significant enough to have solved the problem but I consider the TS Baseline are already at that stage because it was from a production product.

1.6 V. Conclusion

1.6.1 Free-Form Visualization

In [51]:
```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
             precision    recall  f1-score   support

          0       0.98      0.92      0.95      2676
          1       1.00      0.88      0.93       275
          2       1.00      0.98      0.99      8895
          3       0.98      0.97      0.98      5489
          4       0.58      0.21      0.31        33
          5       0.94      0.93      0.94     31956
          6       0.97      0.89      0.93     16022
          7       0.71      0.35      0.47       341
          8       0.92      0.92      0.92     33276
          9       0.94      0.88      0.91      7192
         10       1.00      0.82      0.90       385
         11       0.93      0.91      0.92     31241
         12       0.99      0.97      0.98      6118
         13       0.97      0.88      0.93       998
         14       0.99      0.95      0.97      4674
         15       0.88      0.48      0.62        46
```

```
             16       0.98      0.70      0.82        80
             17       1.00      0.94      0.97       307
             18       1.00      0.98      0.99      1257
             19       0.99      0.97      0.98      3050
             20       1.00      0.99      0.99      2725
             21       1.00      0.98      0.99       439
             22       1.00      0.94      0.97      7283
             23       1.00      0.99      0.99      1041
             24       1.00      0.95      0.97      4713
             25       1.00      0.96      0.98      4070
             26       1.00      0.96      0.98      4076
             27       0.99      0.85      0.92     13394
             28       0.99      0.95      0.97     10120
             29       0.98      0.93      0.95     12594
             30       1.00      0.94      0.97     23006
             31       0.98      0.98      0.98    238156

avg / total       0.97      0.95      0.96    475928
```

The classification report shows that besides not beating the benchmark, the f1-score of good part of the labels are good enough to enrich the data of the text block.

1.6.2 Reflection

The process of participating in a Kaggle competition, download a dataset, explore the data, explore the data with visualization, do some data preprocessing for the dataset, trying some different algorithms, iterating on the different techniques was an amazing experience and give me a boost on confidence that someday I can be a machine learning engineer, I thank you Udacity for this amazing course. This process was very interesting and I learn a lot with, one of the main learnings was that RAM memory can be a huge problem, a number of times have to reboot my notebook because the dataset and the transformations of the data fill up the memory and the swap freezing my notebook. Another interesting thing is the dataset of the competition having 140+ features and 33 labels and 1.7 million rows but with no explanation of what is each feature and label, so I have to train an algorithm more generic to treat this. I learn a lot doing this solution and this was my real expectation because I know it would be a hard task. The proposed solution can be used on similar problems and be of great value.

1.6.3 Improvement

There are techniques that could be made on the proposed solution to get a better result as try to derive some relationship between the content features or use an online learning model. Others algorithms that I research but did not implement are deep learning neural networks, some embeddings to the content features. My solution almost did the benchmark, so for leaderboard of the competition for sure has better solutions than mine and this competition has 4 years, so deep learning techniques can create a new benchmark for this dataset.