Описание данных - https://archive.ics.uci.edu/ml/datasets/banknote+authentication# (https://archive.ics.uci.edu/ml/datasets/banknote+authentication)

In [3]:

```
import pandas as pd
import numpy as np
data = pd.read_csv("car.data", header=None)
data.head(3)
```

Out[3]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| **0** | vhigh | vhigh | 2 | 2 | small | low | unacc |
| **1** | vhigh | vhigh | 2 | 2 | small | med | unacc |
| **2** | vhigh | vhigh | 2 | 2 | small | high | unacc |

In [4]:

```
print(data.shape)
```

```
(1728, 7)
```

In [5]:

```
data.iloc[:, -1].value_counts()
```

Out[5]:

```
unacc    1210
acc       384
good       69
vgood      65
Name: 6, dtype: int64
```

In [6]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1728 entries, 0 to 1727
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   0       1728 non-null   object
 1   1       1728 non-null   object
 2   2       1728 non-null   object
 3   3       1728 non-null   object
 4   4       1728 non-null   object
 5   5       1728 non-null   object
 6   6       1728 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

Разбиваем выборку на тренировочную и тестовую части и обучаем модель (в примере - градиентный

бустинг)

In [7]:

```
data.describe(include='object')
```

Out[7]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| **count** | 1728 | 1728 | 1728 | 1728 | 1728 | 1728 | 1728 |
| **unique** | 4 | 4 | 4 | 3 | 3 | 3 | 4 |
| **top** | high | high | 4 | 4 | small | high | unacc |
| **freq** | 432 | 432 | 432 | 576 | 576 | 576 | 1210 |

In [9]:

```
t = [0,1,2,3,4,5,6]
for tt in t:
    k = 0
    for i in data[tt].unique():
        data.loc[data[tt] == i, tt] = k
        k = k + 1
    data[tt]=  pd.to_numeric(data[tt], errors='coerce')
```

In [10]:

```
CAT_FEATURE_NAMES = [0,1,2,3,4,5,6]

SELECTED_FEATURE_NAMES =CAT_FEATURE_NAMES
```

In [11]:

```
%matplotlib inline
import matplotlib.pyplot as plt
```

In [14]:

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

In [15]:

```
from sklearn.model_selection import train_test_split

x_data = data.iloc[:,:-1]
y_data = data.iloc[:,-1]

x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.2, random_s
```

In [25]:

```
import xgboost as xgb

model = xgb.XGBClassifier()

model.fit(x_train, y_train)
y_predict = model.predict(x_test)
```

```
C:\Users\voron\AppData\Roaming\Python\Python37\site-packages\xgboost\sklear
n.py:1146: UserWarning: The use of label encoder in XGBClassifier is depreca
ted and will be removed in a future release. To remove this warning, do the
following: 1) Pass option use_label_encoder=False when constructing XGBClass
ifier object; and 2) Encode your labels (y) as integers starting with 0, i.
e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[19:53:16] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'multi:softprob' was changed from 'merror' to
'mlogloss'. Explicitly set eval_metric if you'd like to restore the old beha
vior.
```

In [40]:

```
from sklearn.metrics import recall_score, precision_score, roc_auc_score, accuracy_score, f

def evaluate_results(y_test, y_predict):
    print('Classification results:')
    f1 = f1_score(y_test, y_predict, average='micro')
    print("f1: %.2f%%" % (f1 * 100.0))
#     roc = roc_auc_score(y_test, y_predict,multi_class = 'ovr' )
#     print("roc: %.2f%%" % (roc * 100.0))
    rec = recall_score(y_test, y_predict, average='micro')
    print("recall: %.2f%%" % (rec * 100.0))
    prc = precision_score(y_test, y_predict, average='macro')
    print("precision: %.2f%%" % (prc * 100.0))


evaluate_results(y_test, y_predict)
```

```
Classification results:
f1: 100.00%
recall: 100.00%
precision: 100.00%
```

# Теперь очередь за PU learning¶

In [41]:

```
mod_data = data.copy()
#get the indices of the positives samples
pos_ind = np.where(mod_data.iloc[:,-1].values == 1)[0]
#shuffle them
np.random.shuffle(pos_ind)
# leave just 20% of the positives marked
pos_sample_len = int(np.ceil(0.6 * len(pos_ind)))
print(f'Using {pos_sample_len}/{len(pos_ind)} as positives and unlabeling the rest')
pos_sample = pos_ind[:pos_sample_len]
```

```
Using 231/384 as positives and unlabeling the rest
```

In [42]:

```
mod_data['class_test'] = -1
mod_data.loc[pos_sample,'class_test'] = 1
print('target variable:\n', mod_data.iloc[:,-1].value_counts())
```

```
target variable:
 -1    1497
 1      231
Name: class_test, dtype: int64
```

In [43]:

```
mod_data.head(10)
```

Out[43]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | class_test |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 |
| **1** | 0 | 0 | 0 | 0 | 0 | 1 | 0 | -1 |
| **2** | 0 | 0 | 0 | 0 | 0 | 2 | 0 | -1 |
| **3** | 0 | 0 | 0 | 0 | 1 | 0 | 0 | -1 |
| **4** | 0 | 0 | 0 | 0 | 1 | 1 | 0 | -1 |
| **5** | 0 | 0 | 0 | 0 | 1 | 2 | 0 | -1 |
| **6** | 0 | 0 | 0 | 0 | 2 | 0 | 0 | -1 |
| **7** | 0 | 0 | 0 | 0 | 2 | 1 | 0 | -1 |
| **8** | 0 | 0 | 0 | 0 | 2 | 2 | 0 | -1 |
| **9** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | -1 |

In [44]:

```
mod_data['class_test'].value_counts()
```

Out[44]:

```
-1    1497
 1     231
Name: class_test, dtype: int64
```

In [45]:

```
x_data = mod_data.iloc[:,:-2].values # just the X
y_labeled = mod_data.iloc[:,-1].values # new class (just the P & U)
y_positive = mod_data.iloc[:,-2].values # original class
```

In [46]:

```
mod_data = mod_data.sample(frac=1)
neg_sample = mod_data[mod_data['class_test']==-1][:len(mod_data[mod_data['class_test']==1])
sample_test = mod_data[mod_data['class_test']==-1][len(mod_data[mod_data['class_test']==1])
pos_sample = mod_data[mod_data['class_test']==1]
print(neg_sample.shape, pos_sample.shape)
sample_train = pd.concat([neg_sample, pos_sample]).sample(frac=1)
```

```
(231, 8) (231, 8)
```

In [47]:

```
model = xgb.XGBClassifier()

model.fit(sample_train.iloc[:,:-2].values,
          sample_train.iloc[:,-2].values)
y_predict = model.predict(sample_test.iloc[:,:-2].values)
evaluate_results(sample_test.iloc[:,-2].values, y_predict)
```

```
C:\Users\voron\AppData\Roaming\Python\Python37\site-packages\xgboost\sklear
n.py:1146: UserWarning: The use of label encoder in XGBClassifier is depreca
ted and will be removed in a future release. To remove this warning, do the
following: 1) Pass option use_label_encoder=False when constructing XGBClass
ifier object; and 2) Encode your labels (y) as integers starting with 0, i.
e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[20:02:20] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'multi:softprob' was changed from 'merror' to
'mlogloss'. Explicitly set eval_metric if you'd like to restore the old beha
vior.
Classification results:
f1: 93.29%
recall: 93.29%
precision: 84.88%
```

In [ ]:

```

```