In [13]:

```
#This project predicts whether a customer will repay their credit within 90 day
s. The tasks and dataset were posted by Yury Kashnitsky on Kaggle.
import warnings
warnings.filterwarnings('ignore')

import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from matplotlib import rcParams
rcParams['figure.figsize'] = 7, 5
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier


#for each colums, replacing NaNs with the column median
def fill_nan(table):
    for col in table.columns:
        table[col] = table[col].fillna(table[col].median())
    return table

data = pd.read_csv('/Users/iqra/Desktop/credit_scoring_sample.csv')
data.head()
```

Out[13]:

| | SeriousDlqin2yrs | age | NumberOfTime30-59DaysPastDueNotWorse | DebtRatio | NumberOfTimes90DaysLate |
|---|---|---|---|---|---|
| 0 | 0 | 64 | 0 | 0.249908 | 0 |
| 1 | 0 | 58 | 0 | 3870.000000 | 0 |
| 2 | 0 | 41 | 0 | 0.456127 | 0 |
| 3 | 0 | 43 | 0 | 0.000190 | 0 |
| 4 | 1 | 49 | 0 | 0.271820 | 0 |

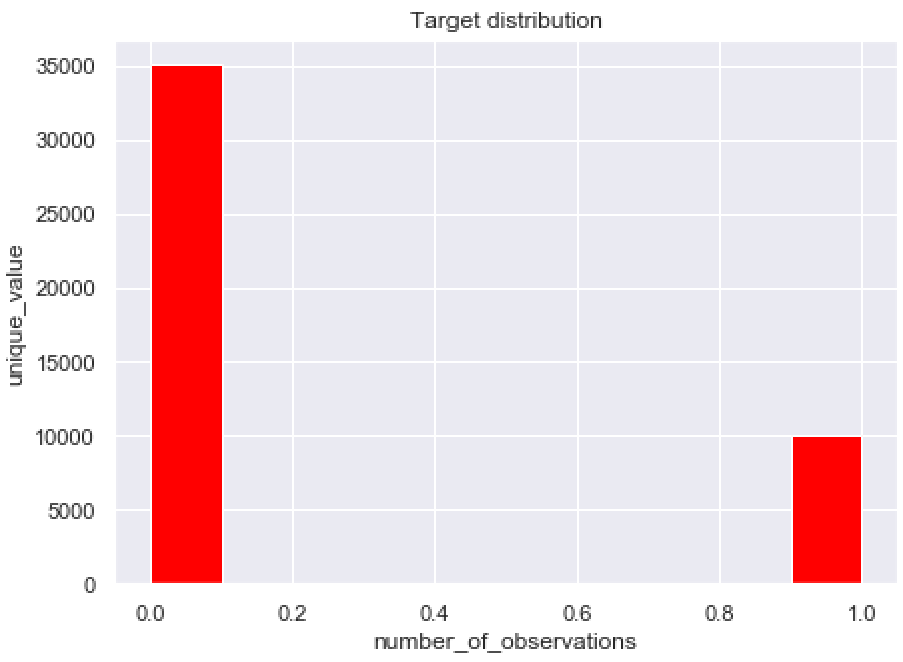
In [14]:

```
ax = data['SeriousDlqin2yrs'].hist(orientation='vertical', color='red')
ax.set_xlabel("number_of_observations")
ax.set_ylabel("unique_value")
ax.set_title("Target distribution")

print('Distribution of the target:')
data['SeriousDlqin2yrs'].value_counts()/data.shape[0]

independent_columns_names = [x for x in data if x != 'SeriousDlqin2yrs']
independent_columns_names
#visualizing the target distribution
table = fill_nan(data)
X = table[independent_columns_names]
y = table['SeriousDlqin2yrs']
```

```
Distribution of the target:
```



In [29]:

```
#Logistic Regression
lr = LogisticRegression(random_state=5, class_weight='balanced')
#finding the best regulation coefficient for our model
parameters = {'C': (0.0001, 0.001, 0.01, 0.1, 1, 10)}
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=5)
grid_search = GridSearchCV(lr, parameters,  scoring='roc_auc', cv=skf)
grid_search = grid_search.fit(X, y)
grid_search.best_estimator_
grid_search.best_score_


lr = LogisticRegression(C=0.001, random_state=5, class_weight='balanced')
scal = StandardScaler()
lr.fit(scal.fit_transform(X), y)
```

Out[29]:

```
LogisticRegression(C=0.001, class_weight='balanced', random_state=5)
```

In [40]:

```
pd.DataFrame({'feat': independent_columns_names,
              'coef': lr.coef_.flatten().tolist()}).sort_values(by='coef', ascen
ding=False)
#creating a random forest function
rf = RandomForestClassifier(n_estimators=100, n_jobs=-1, random_state=42,
                            class_weight='balanced')
#searching for the best params
parameters = {'max_features': [1, 2, 4], 'min_samples_leaf': [3, 5, 7, 9], 'max_
depth': [5,10,15]}
rf_grid_search = GridSearchCV(rf, parameters, n_jobs=-1, scoring='roc_auc', cv=s
kf, verbose=True)
rf_grid_search = rf_grid_search.fit(X, y)
```

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent w
orkers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:   53.4s
[Parallel(n_jobs=-1)]: Done 180 out of 180 | elapsed:  5.6min finish
ed

0.026820670654852385
```

In [25]:

```
#setting up params for bagging
from sklearn.ensemble import BaggingClassifier
from sklearn.model_selection import cross_val_score, RandomizedSearchCV
parameters = {'max_features': [2, 3, 4], 'max_samples': [0.5, 0.7, 0.9],
              'base_estimator__C': [0.0001, 0.001, 0.01, 1, 10, 100]}

#we will use logistic regression for its ability to better interpret and evaluat
e features
bg = BaggingClassifier(LogisticRegression(class_weight='balanced'),
                        n_estimators=100, n_jobs=-1, random_state=42)
r_grid_search = RandomizedSearchCV(bg, parameters, n_jobs=-1,
                                   scoring='roc_auc', cv=skf, n_iter=20, random_
state=1,
                                   verbose=True)
r_grid_search = r_grid_search.fit(X, y)

#finding the ROC AUC
r_grid_search.best_score_
r_grid_search.best_estimator_
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent w
orkers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  5.2min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed: 13.1min finish
ed
```

Out[25]:

```
BaggingClassifier(base_estimator=LogisticRegression(C=100,
                                                    class_weight='ba
lanced'),
                  max_features=2, max_samples=0.9, n_estimators=100,
n_jobs=-1,
                  random_state=42)
```

In [ ]: