РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра информационных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

Дисциплина: Интеллектуальный анализ данных

Студент: Николаев Александр Викторович

Группа: НФИбд-01-17

Москва 2020

Вариант №5

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

1. Используя функционал библиотеки Pandas, считаем заданный набор данных из репозитария UCI.

https://archive.ics.uci.edu/ml/machine-learning-databases/credit-screening/crx.data (https://archive.ics.uci.edu/ml/machine-learning-databases/credit-screening/crx.data)

In [2]:

```
data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/credit-screening/crx.data',
                   header=None, prefix='A')
```

1. Проведем исследование набора данных

In [3]:

```
data.head()
```

Out[3]:

	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15
0	b	30.83	0.000	u	g	w	v	1.25	t	t	1	f	g	00202	0	+
1	a	58.67	4.460	u	g	q	h	3.04	t	t	6	f	g	00043	560	+
2	a	24.50	0.500	u	g	q	h	1.50	t	f	0	f	g	00280	824	+
3	b	27.83	1.540	u	g	w	v	3.75	t	t	5	t	g	00100	3	+
4	b	20.17	5.625	u	g	w	v	1.71	t	f	0	f	s	00120	0	+

Числовые признаки: `A1` , `A2` , `A7` , `A10` , `A13` , `A14`

Категориальные признаки: `A0` , `A3` , `A4` , `A5` , `A6` , `A8` , `A9` , `A11` , `A12`

Таргет: `A15`

In [4]:

```
num_feats = ['A1', 'A2', 'A7', 'A10', 'A13', 'A14']
cat_feats = ['A0', 'A3', 'A4', 'A5', 'A6', 'A8', 'A9', 'A11', 'A12']
target = 'A15'
```

Заполним пропуски ('?') в числовых признаках медианным значением, а в категориальных - модой.

In [5]:

```
for feat in num_feats:
    data[feat] = data[feat].replace('?', np.nan).astype(float)
```

In [6]:

```
data.replace('?', np.nan, inplace=True)

print(f'Количество пропусков до заполнения: {data.isna().sum().sum()}')

data[num_feats] = data[num_feats].fillna(data[num_feats].median())
data[cat_feats] = data[cat_feats].fillna(data[cat_feats].mode().squeeze())

print(f'Количество пропусков после заполнения: {data.isna().sum().sum()}')
```

```
Количество пропусков до заполнения: 67
Количество пропусков после заполнения: 0
```

1. Определим признак, содержащий метку класса. Если признак, содержащий метку класса, принимает более 10 различных значений, то выполним дискретизацию этого признака, перейдя к 3-4 диапазонам значений.

Метку содержит признак A15

Переименуем его в target

In [7]:

```
data.rename(columns={'A15': 'target'}, inplace=True)
# data['target'].replace({'-': -1, '+': 1}, inplace=True)
print(data['target'].value_counts())
```

```
-    383
+    307
Name: target, dtype: int64
```

Поскольку целевой признак имеет всего две метки класса, то дискретизация не требуется.

1. Определим числовой признак, имеющий максимальную дисперсию. Исследуем, принимает ли это признак дискретные или непрерывные значения.

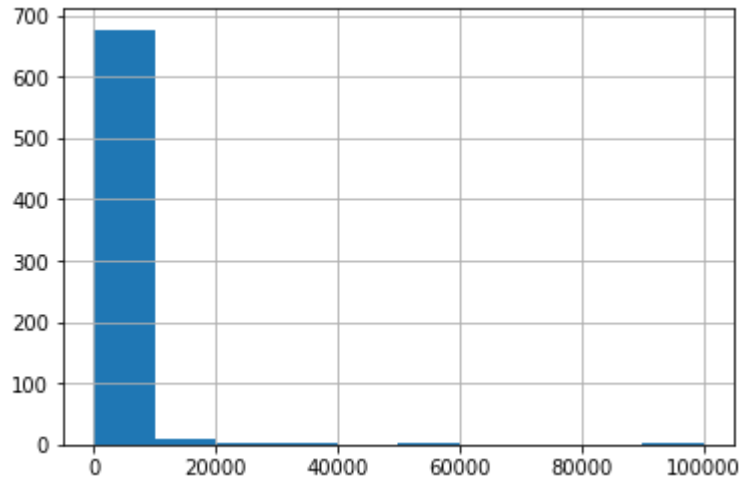
In [8]:

```
max_var_feat = num_feats[data[num_feats].var().argmax()]
print(f'Числовой признак с наибольшей дисперсией: {max_var_feat}, дисперсия = {data[num
_feats].var().max()}')
```

```
Числовой признак с наибольшей дисперсией: A14, дисперсия = 27145169.084840
517
```

In [9]:

```
data[max_var_feat].hist();
```



In [10]:

```
data[max_var_feat].value_counts()
```

Out[10]:

```
0.0        295
1.0         29
500.0       10
1000.0      10
2.0          9
          ...
1349.0       1
2206.0       1
51100.0      1
4071.0       1
10000.0      1
Name: A14, Length: 240, dtype: int64
```

Можно сказать, что признак принимает дискретные значения, т.к.: 1) нет действительных значений; 2) уникальных значение всего 240, что сильно меньше размера датасета; 3) Гистограмма признака показывает не непрерывное распределение

1. При помощи класса SelectKBest библиотеки scikit-learn найдём два признака, имеющих наиболее выраженную взаимосвязь с признаком, имеющим максимальную дисперсию.

Добавим в исследование категории, предварительно закодировав их с помощью LabelEncoding

In [11]:

```
for feat in cat_feats:
    data[feat] = data[feat].astype('category').cat.codes
```

In [12]:

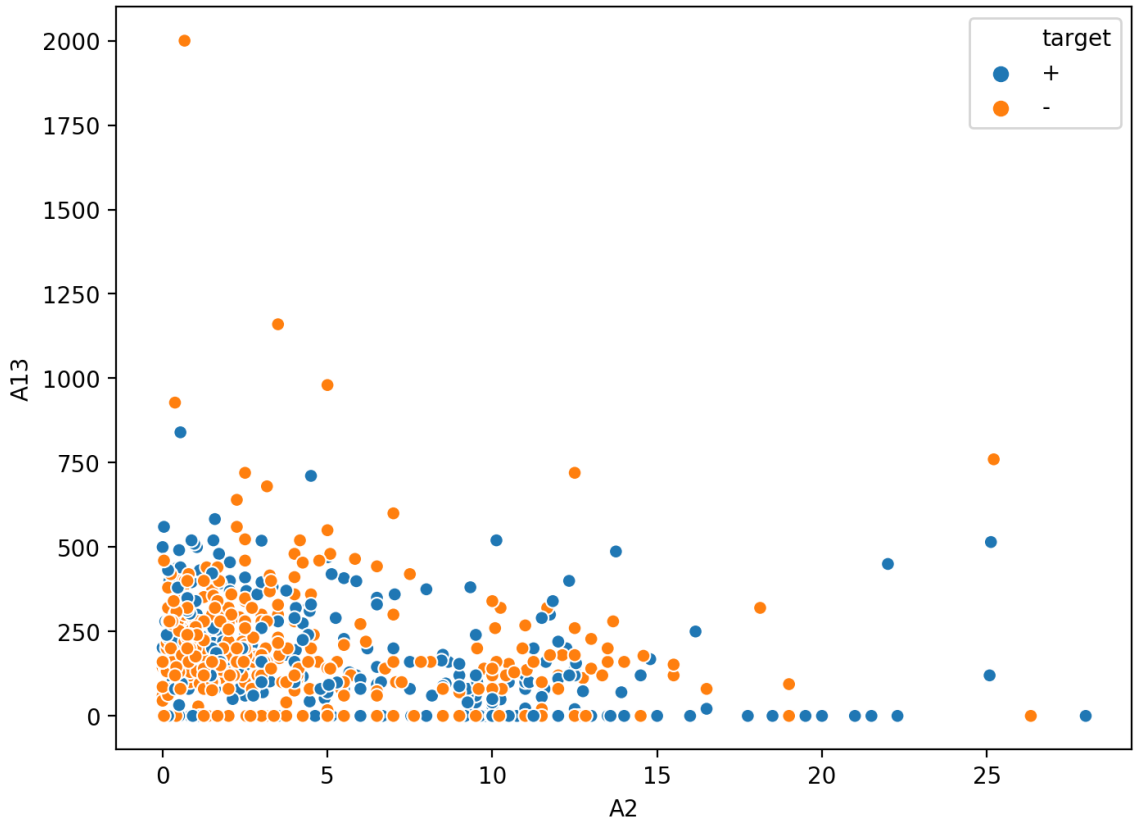
```
from sklearn.feature_selection import SelectKBest, f_regression

X = data[num_feats].drop(columns=max_var_feat)
y = data[max_var_feat]

k_best = SelectKBest(score_func=f_regression, k=2)
fit = k_best.fit(X, y)

print("\nОценки признаков:\n")
for col, score in zip(X.columns, fit.scores_):
    print(f'{col} : {score}')


cols = k_best.get_support(indices=True)
print("\nОтобранные признаки:\n", X.iloc[:, cols].head())
```

```
Оценки признаков:

A1 : 0.2411530061137094
A2 : 10.5897958111878
A7 : 1.8185699022332564
A10 : 2.80239661507955
A13 : 2.905348612910606

Отобранные признаки:
       A2    A13
0  0.000  202.0
1  4.460   43.0
2  0.500  280.0
3  1.540  100.0
4  5.625  120.0
```

1. Визуализируем набор данных в виде точек плоскости с координатами, соответствующими найденным признакам, отображая точки различных классов разными цветами. Подпишим оси и рисунок, создадим легенду набора данных.

In [13]:

```
plt.figure( figsize=(8, 6), dpi=200 )
sns.scatterplot(x=data['A2'], y=data['A13'], hue=data['target']);
```



1. Оставляя в наборе данных только числовые признаки, найдём и выведем на экран размерность метода главных компонент (параметр n_components), для которой доля объясняемой дисперсии будет не менее 99%.

In [14]:

```
from sklearn.decomposition import PCA
```

In [15]:

```
X = data[num_feats]

for r in range(1, len(num_feats)+1):
    pca = PCA(n_components=r)
    pca.fit(X)
    print(f"r = {r}\tДисперсия = {sum(pca.explained_variance_ratio_)*100}%")
```

```
r = 1   Дисперсия = 99.89061858357638%
r = 2   Дисперсия = 99.99927484289051%
r = 3   Дисперсия = 99.99980306568906%
r = 4   Дисперсия = 99.9999063932179%
r = 5   Дисперсия = 99.9999706486177%
r = 6   Дисперсия = 99.99999999999997%
```

Для любого количества компонент, доля объясняемой дисперсии будет не менее 99%

1. Пользуясь методом главных компонент, снизим размерность набора данных до двух признаков и изобразим полученный набор данных в виде точек на плоскости, отображая точки различных классов разными цветами. Подпишим оси и рисунок, создадим легенду набора данных.

In [16]:

```
pca = PCA(n_components=2)
pcad = pca.fit_transform(X)

plt.figure(figsize=(8, 6), dpi=200)
sns.scatterplot(x=pcad[:, 0], y=pcad[:, 1], hue=data['target']);
```