Лабораторная работа №1

Махмадзиеев Али 181-331

Задание 1.1 [до 3 баллов].

Подобрать набор данных, подготовленный авторитетным источником, для решения задач информационной безопасности:

«Making it easier to discover datasets».

Изучить способ сбора данных, описание классов и их атрибутов – представить датасет.

Описать метки и атрибуты/признаки, разделить последние на количественные и качественные.

Задание 1.2 [до 5 баллов].

Исследовать (программно) на качественную однородность значения каждого количественного признака.

1. Построить гистограмму: см. «Математика и информатика для юристов».

2. Вычислить значения признака, соответствующие локальным минимумам плотности частоты h.

3. По вычисленным значениям разбить область значений признака на непересекающиеся области однородности.

4. Каждую область однородности представить математическим ожиданием μ и

границами доверительного интервала ±3σ.

Задание 1.3 [до 2 баллов].

Для каждого качественного признака по частотам

значений построить столбчатую диаграмму.

Дата сет «СЛИТЫЕ БИТКОИН ТРАНЗАКЦИИ»

Ссылка: https://www.kaggle.com/xblock/mtgox-leaked-transaction

```
B [42]: import pandas as pd
        import numpy as np

B [43]: import matplotlib.pyplot as plt

B [44]: df = pd.read_csv('DfAli2.csv')
        df
```

Out[44]:

	Source	Target	Trade_Id	Bitcoins	Money	Money_Rate	Date	label
0	586159	100349	1373958491820869	1.250628	124.06234	99.200000	2013-07-16 07:08:11	0
1	199328	115248	1358039479966822	0.054551	0.77872	14.275064	2013-01-13 01:11:20	0
2	105211	96376	1329371016524489	13.543310	56.09165	4.141650	2012-02-16 05:43:36	0
3	69334	320942	1365522594463088	1.074300	239.99089	223.392805	2013-04-09 15:49:55	0
4	89169	28388	1322881924264838	8.443058	25.93792	3.072100	2011-12-03 03:12:04	0
...	...	...	...	...	...	...	...	...
67746	67321	31219	1322472062155919	4.840776	12.16971	2.514000	2011-11-28 09:21:02	0
67747	231	499498	1370529976369716	1.267475	153.61795	121.199951	2013-06-06 14:46:16	0
67748	THK	527401	1378329329034856	13.944896	188256.09400	13500.000031	2013-09-04 21:15:29	1
67749	90128	36865	1345470700972834	0.021053	0.19147	9.094549	2012-08-20 13:51:40	0
67750	231	235134	1354578128349347	0.010000	0.12660	12.660000	2012-12-03 23:42:08	0

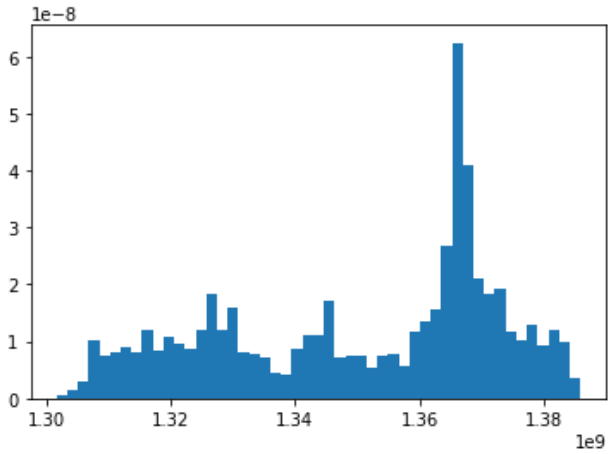
67751 rows × 8 columns

1. Source/КАЧЕСТВЕНЫЙ/Пользователь, продающий биткойны
2. Target/КАЧЕСТВЕНЫЙ/Пользователь, покупающий биткойны
3. Trade_Id/КАЧЕСТВЕНЫЙ/ Идентификатор текущей сделки
4. Bitcoins/КОЛИЧЕСТВЕННЫЙ/ Количество биткойнов, задействованных в текущей транзакции
5. Money/КОЛИЧЕСТВЕННЫЙ/ Долларов, потраченных на покупку биткойнов
6. Money_rate/КОЛИЧЕСТВЕННЫЙ/ Цена за биткойн
7. Date/КОЛИЧЕСТВЕННЫЙ/ Дата транзакции
8. label

Диаграммы количественных признаков

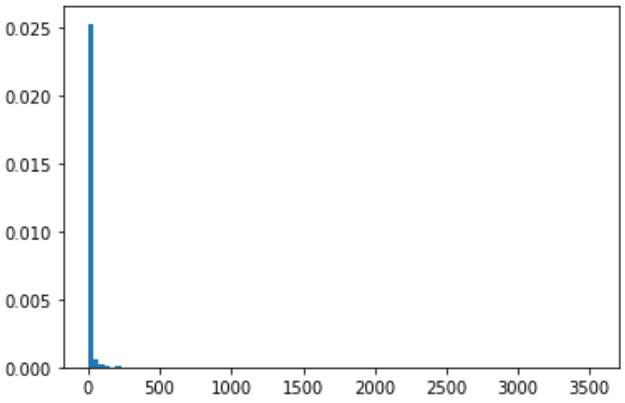
Date:

```
B [10]: plt.hist(df['Date'], bins=k, density=True)
        plt.show()
```



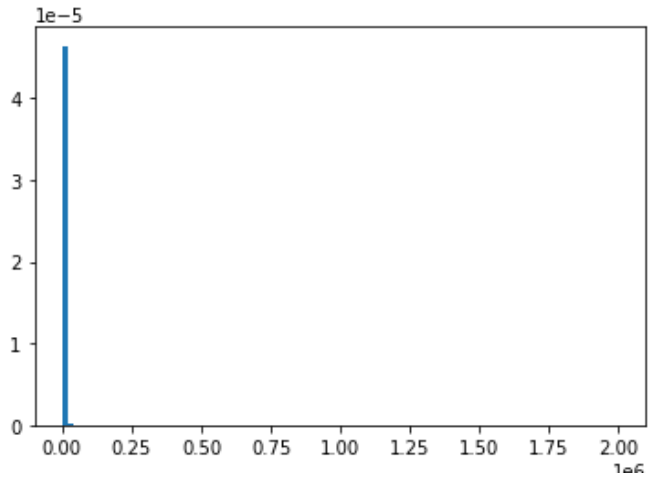
Bitcoins:

```
B [11]: plt.hist(df['Bitcoins'], bins=k2, density=True)
        plt.show()
```



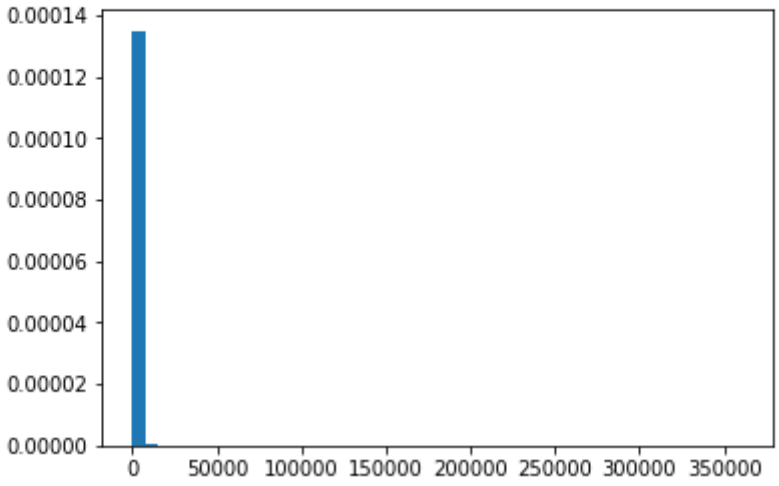
Money:

```
B [12]: plt.hist(df['Money'], bins=k2, density=True)
        plt.show()
```



Money_Rate

```
B [13]: plt.hist(df['Money_Rate'], bins=k, density=True)
        plt.show()
```



Локальные минимумы, области однородности, мат ожидания и доверительные интервалы для количественных признаков:

```
B [33]: # Области однородности для Date
Oblast1Odnorodnosti1_Date = []
Oblast1Odnorodnosti2_Date = []
Oblast1Odnorodnosti3_Date = []
Oblast1Odnorodnosti4_Date = []
Oblast1Odnorodnosti5_Date = []
Oblast1Odnorodnosti6_Date = []
Oblast1Odnorodnosti7_Date = []
Oblast1Odnorodnosti8_Date = []
Oblast1Odnorodnosti9_Date = []
Oblast1Odnorodnosti10_Date = []
Oblast1Odnorodnosti11_Date = []
Oblast1Odnorodnosti12_Date = []
Oblast1Odnorodnosti13_Date = []


for oblast1 in df['Date']:
    if oblast1 <= 1309400000:
        Oblast1Odnorodnosti1_Date.append(oblast1)
    if oblast1 > 1309400000 and oblast1 <= 1314560000:
        Oblast1Odnorodnosti2_Date.append(oblast1)
    if oblast1 > 1314560000 and oblast1 <= 1317985000:
        Oblast1Odnorodnosti3_Date.append(oblast1)
    if oblast1 > 1317985000 and oblast1 <= 1323130000:
        Oblast1Odnorodnosti4_Date.append(oblast1)
    if oblast1 > 1323130000 and oblast1 <= 1328295000:
        Oblast1Odnorodnosti5_Date.append(oblast1)
    if oblast1 > 1328295000 and oblast1 <= 1338560000:
        Oblast1Odnorodnosti6_Date.append(oblast1)
    if oblast1 > 1338560000 and oblast1 <= 1347185000:
        Oblast1Odnorodnosti7_Date.append(oblast1)
    if oblast1 > 1347185000 and oblast1 <= 1352335000:
        Oblast1Odnorodnosti8_Date.append(oblast1)
    if oblast1 > 1352335000 and oblast1 <= 1357465000:
        Oblast1Odnorodnosti9_Date.append(oblast1)
    if oblast1 > 1357465000 and oblast1 <= 1371225000:
        Oblast1Odnorodnosti10_Date.append(oblast1)
    if oblast1 > 1371225000 and oblast1 <= 1376380000:
```

```
B [34]: # Математическое ожидание для Date
mathOzd1_Date = np.mean(Oblast1Odnorodnosti1_Date)
print(mathOzd1_Date, "MATHOZD1_Date")
mathOzd2_Date = np.mean(Oblast1Odnorodnosti2_Date)
print(mathOzd2_Date, "MATHOZD1_Date")
mathOzd3_Date = np.mean(Oblast1Odnorodnosti3_Date)
print(mathOzd3_Date, "Date")
mathOzd4_Date = np.mean(Oblast1Odnorodnosti4_Date)
print(mathOzd4_Date, "Date")
mathOzd5_Date = np.mean(Oblast1Odnorodnosti5_Date)
print(mathOzd5_Date, "Date")
mathOzd6_Date = np.mean(Oblast1Odnorodnosti6_Date)
print(mathOzd6_Date, "Date")
mathOzd7_Date = np.mean(Oblast1Odnorodnosti7_Date)
print(mathOzd7_Date, "Date")
mathOzd8_Date = np.mean(Oblast1Odnorodnosti8_Date)
print(mathOzd8_Date, "Date")
mathOzd9_Date = np.mean(Oblast1Odnorodnosti9_Date)
print(mathOzd9_Date, "Date")
mathOzd10_Date = np.mean(Oblast1Odnorodnosti10_Date)
print(mathOzd10_Date, "Date")
mathOzd11_Date = np.mean(Oblast1Odnorodnosti11_Date)
print(mathOzd11_Date, "Date")
mathOzd12_Date = np.mean(Oblast1Odnorodnosti12_Date)
print(mathOzd12_Date, "Date")
mathOzd13_Date = np.mean(Oblast1Odnorodnosti13_Date)
print(mathOzd13_Date, "Date")
```

```python
# Среднеквадратическое отклонение для Date
std1_Date = np.std(Oblast1Odnorodnosti1_Date)
print(std1_Date)
std2_Date = np.std(Oblast1Odnorodnosti2_Date)
print(std2_Date)
std3_Date = np.std(Oblast1Odnorodnosti3_Date)
print(std3_Date)
std4_Date = np.std(Oblast1Odnorodnosti4_Date)
print(std4_Date)
std5_Date = np.std(Oblast1Odnorodnosti5_Date)
print(std5_Date)
std6_Date = np.std(Oblast1Odnorodnosti6_Date)
print(std6_Date)
std7_Date = np.std(Oblast1Odnorodnosti7_Date)
print(std7_Date)
std8_Date = np.std(Oblast1Odnorodnosti8_Date)
print(std8_Date)
std9_Date = np.std(Oblast1Odnorodnosti9_Date)
print(std9_Date)
std10_Date = np.std(Oblast1Odnorodnosti10_Date)
print(std10_Date)
std11_Date = np.std(Oblast1Odnorodnosti11_Date)
print(std11_Date)
std12_Date = np.std(Oblast1Odnorodnosti12_Date)
print(std12_Date)
std13_Date = np.std(Oblast1Odnorodnosti13_Date)
print(std13_Date)
```

```python
# Отрицательный доверительный интервал для Date
otr1cateln1_doverit_int_Date_1 = mathOzd1_Date - 3 * std1_Date
print(otr1cateln1_doverit_int_Date_1)
otr1cateln1_doverit_int_Date_2 = mathOzd2_Date - 3 * std2_Date
print(otr1cateln1_doverit_int_Date_2)
otr1cateln1_doverit_int_Date_3 = mathOzd3_Date - 3 * std3_Date
print(otr1cateln1_doverit_int_Date_3)
otr1cateln1_doverit_int_Date_4 = mathOzd4_Date - 3 * std4_Date
print(otr1cateln1_doverit_int_Date_4)
otr1cateln1_doverit_int_Date_5 = mathOzd5_Date - 3 * std5_Date
print(otr1cateln1_doverit_int_Date_5)
otr1cateln1_doverit_int_Date_6 = mathOzd6_Date - 3 * std6_Date
print(otr1cateln1_doverit_int_Date_6)
otr1cateln1_doverit_int_Date_7 = mathOzd7_Date - 3 * std7_Date
print(otr1cateln1_doverit_int_Date_7)
otr1cateln1_doverit_int_Date_8 = mathOzd8_Date - 3 * std8_Date
print(otr1cateln1_doverit_int_Date_8)
otr1cateln1_doverit_int_Date_9 = mathOzd9_Date - 3 * std9_Date
print(otr1cateln1_doverit_int_Date_9)
otr1cateln1_doverit_int_Date_10 = mathOzd10_Date - 3 * std10_Date
print(otr1cateln1_doverit_int_Date_10)
otr1cateln1_doverit_int_Date_11 = mathOzd11_Date - 3 * std11_Date
print(otr1cateln1_doverit_int_Date_11)
otr1cateln1_doverit_int_Date_12 = mathOzd12_Date - 3 * std12_Date
print(otr1cateln1_doverit_int_Date_12)
otr1cateln1_doverit_int_Date_13 = mathOzd13_Date - 3 * std13_Date
print(otr1cateln1_doverit_int_Date_13)
```

```python
# Положительный доверительный интервал для Date
polozhit_dover1telni_int_Date_1 = mathOzd1_Date +  3 * std1_Date
print(polozhit_dover1telni_int_Date_1)
polozhit_dover1telni_int_Date_2 = mathOzd2_Date +  3 * std2_Date
print(polozhit_dover1telni_int_Date_2)
polozhit_dover1telni_int_Date_3 = mathOzd2_Date +  3 * std2_Date
print(polozhit_dover1telni_int_Date_3)
polozhit_dover1telni_int_Date_4 = mathOzd2_Date +  3 * std2_Date
print(polozhit_dover1telni_int_Date_4)
polozhit_dover1telni_int_Date_5 = mathOzd2_Date +  3 * std2_Date
print(polozhit_dover1telni_int_Date_5)
polozhit_dover1telni_int_Date_6 = mathOzd2_Date +  3 * std2_Date
print(polozhit_dover1telni_int_Date_6)
polozhit_dover1telni_int_Date_7 = mathOzd2_Date +  3 * std2_Date
print(polozhit_dover1telni_int_Date_7)
polozhit_dover1telni_int_Date_8 = mathOzd2_Date +  3 * std2_Date
print(polozhit_dover1telni_int_Date_8)
polozhit_dover1telni_int_Date_9 = mathOzd2_Date +  3 * std2_Date
print(polozhit_dover1telni_int_Date_9)
polozhit_dover1telni_int_Date_10 = mathOzd2_Date +  3 * std2_Date
print(polozhit_dover1telni_int_Date_10)
polozhit_dover1telni_int_Date_11 = mathOzd2_Date +  3 * std2_Date
print(polozhit_dover1telni_int_Date_11)
polozhit_dover1telni_int_Date_12 = mathOzd2_Date +  3 * std2_Date
print(polozhit_dover1telni_int_Date_12)
polozhit_dover1telni_int_Date_13 = mathOzd2_Date +  3 * std2_Date
print(polozhit_dover1telni_int_Date_13)
```

В [55]:
```
# Области однородности для Bitcoins
Oblast1Odnorodnosti1_Bitcoins = []
Oblast1Odnorodnosti2_Bitcoins = []


for oblast1 in df['Bitcoins']:
    if oblast1 <= 170.1:
        Oblast1Odnorodnosti1_Bitcoins.append(oblast1)
    if oblast1 > 170.1:
        Oblast1Odnorodnosti2_Bitcoins.append(oblast1)
```

В [56]:
```
# Математическое ожидание для Bitcoins
mathOzd1_Bitcoins = np.mean(Oblast1Odnorodnosti1_Bitcoins)
print(mathOzd1_Date, "MATHOZD1_Date")
mathOzd2_Bitcoins = np.mean(Oblast1Odnorodnosti2_Bitcoins)
print(mathOzd2_Date, "MATHOZD1_Date")
```

```
1307055875.2645693 MATHOZD1_Date
1311801098.5062883 MATHOZD1_Date
```

В [57]:
```
# Среднеквадратическое отклонение для Bitcoins
std1_Bitcoins = np.std(Oblast1Odnorodnosti1_Bitcoins)
print(std1_Bitcoins)
std2_Bitcoins = np.std(Oblast1Odnorodnosti2_Bitcoins)
print(std2_Bitcoins)
```

```
14.72738976394634
361.0876514179379
```

В [58]:
```
# Отрицательный доверительный интервал для Bitcoins
otr1cateln1_doverit_int_Bitcoins_1 = mathOzd1_Bitcoins - 3 * std1_Bitcoins
print(otr1cateln1_doverit_int_Bitcoins_1)
otr1cateln1_doverit_int_Bitcoins_2 = mathOzd2_Bitcoins - 3 * std2_Bitcoins
print(otr1cateln1_doverit_int_Bitcoins_2)
```

```
-38.71544884672875
-702.1128935362256
```

В [61]:
```
# Всё о Money
mean_SOUD = np.mean(df['Money'])
print(mean_SOUD, "MEAN_Money")
std_SOUD = np.std(df['Money'])
print(std_SOUD, "STD_Money")
otr1cateln1_doverit_int_SOUD = mean_SOUD - 3 * std_SOUD
print(otr1cateln1_doverit_int_SOUD, "Otric_Money")
polozhit_dover1telni_int_SOUD = mean_SOUD +  3 * std_SOUD
print(polozhit_dover1telni_int_SOUD, "Poloj_Money")
```

```
620.2842132089297 MEAN_Money
15203.196606851536 STD_Money
-44989.30560734568 Otric_Money
46229.87403376354 Poloj_Money
```

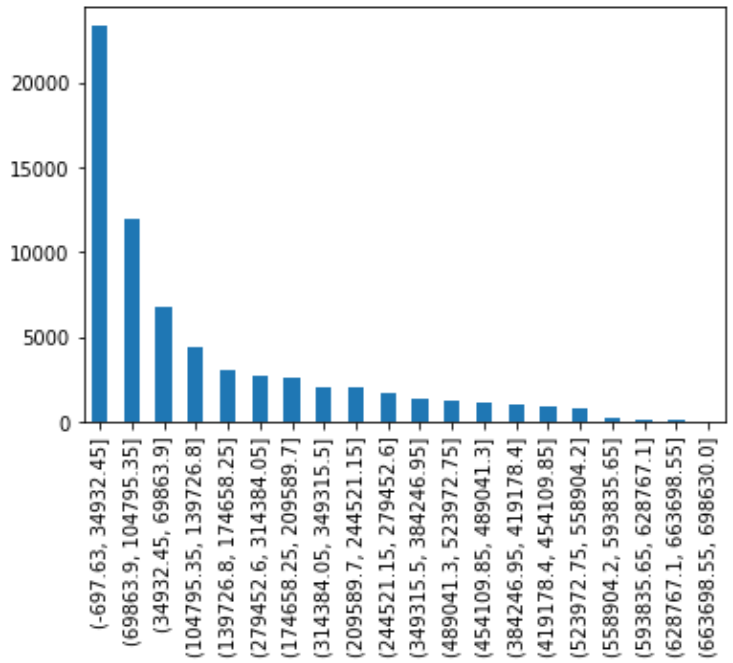
В [63]:
```
# Всё о Money_Rate
mean_SOUD = np.mean(df['Money_Rate'])
print(mean_SOUD, "MEAN_Money_Rate")
std_SOUD = np.std(df['Money_Rate'])
print(std_SOUD, "STD_Money_Rate")
otr1cateln1_doverit_int_SOUD = mean_SOUD - 3 * std_SOUD
print(otr1cateln1_doverit_int_SOUD, "Otric_Money_Rate")
polozhit_dover1telni_int_SOUD = mean_SOUD +  3 * std_SOUD
print(polozhit_dover1telni_int_SOUD, "Poloj_Money_Rate")
```

```
170.12305532229786 MEAN_Money_Rate
2571.996793191856 STD_Money_Rate
-7545.8673242532695 Otric_Money_Rate
7886.113434897865 Poloj_Money_Rate
```

Диаграммы качественных признаков:

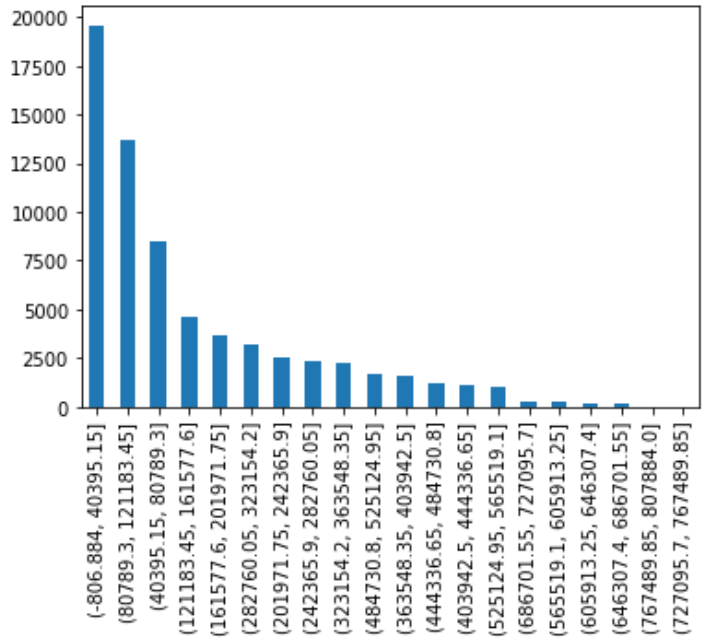
Source:

```
B [89]: plt.figure()
        df['Source'].value_counts(bins=20).plot.bar()
        plt.show()
```



Target:

```
B [90]: plt.figure()
        df['Target'].value_counts(bins=20).plot.bar()
        plt.show()
```



Trade_Id

B [97]:

```
plt.figure()
df['Trade_Id'].value_counts(bins=2).plot.bar()
plt.show()
```