МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
Дальневосточный федеральный университет

ШКОЛА ЕСТЕСТВЕННЫХ НАУК

Кафедра информационной безопасности

О Т Ч Е Т
о прохождении учебной практики (учебно-лабораторного практикума)

Выполнил студент
гр. С8117-10.05.01ммзи
_____ Вяргизова Ю.В.
(подпись)

Отчет защищен с оценкой

_____
_____ С.С. Зотов
(подпись) (И.О. Фамилия)
« 26 » июня 2021 г.

Руководитель практики
Старший преподаватель кафедры
информационной безопасности ШЕН
_____ С.С. Зотов
(подпись) (И.О. Фамилия)

Регистрационный № _____
« 26 » июня 2021 г.

_____ _____
(подпись) (И.О. Фамилия)

Практика пройдена в срок
с « 22 » февраля 2021 г.
по « 26 » июня 2021 г.
на предприятии
Кафедра информационной
безопасности ШЕН ДВФУ

г. Владивосток
2021

Характеристика

Выдана студенту 4 курса, специальности «Компьютерная безопасность», специализации «Математические методы защиты информации», Вяргизовой Юлии Владимировне.

Вяргизова Юлия Владимировна, в период с 22.02.2021 по 26.06.2021 года, проходил учебную практику (учебно-лабораторный практикум) на кафедре информационной безопасности ШЕН ДВФУ.

За время прохождения практики Юлия проявила усердие, тягу к знаниям, огромное желание и трудолюбие, а также неподдельный интерес к изучению материала, требуемого для написания отчета. Приходила на консультацию вовремя с перечнем вопросов, с подробным и исчерпывающим описанием о текущем состоянии практики, со списком отмеченных задач. Внимательно изучала предложенные материалы и литературу на интересующую тематику.

Вяргизова Ю.В. полностью выполнила предусмотренную программу практики, продемонстрировала умения самостоятельно решать практические вопросы, применяя теоретическую базу, полученную в учебный период, а также при самостоятельном обучении.

При выполнении поставленных задач Вяргизова Ю.В. характеризуется инициативностью, сообразительностью и ответственностью.

Старший преподаватель кафедры
информационной безопасности _____ Зотов С.С.

ДНЕВНИК СТУДЕНТА

Дата	Рабочее место	Краткое содержание выполняемых работ	Отметки руководителя
22.02.21 – 27.04.21	КИБ	Выбор темы практической работы	
28.04.21 – 30.04.21	КИБ	Поиск материала	
01.05.21 – 05.05.21	КИБ	Анализ найденного материала	
06.05.21 – 14.06.21	КИБ	Реализация алгоритмов кластеризации	
15.06.21 – 20.06.21	КИБ	Написание отчёта по проделанной работе	
21.06.21 – 26.06.21	КИБ	Сдача готового отчета преподавателю	

Студент _____Вяргизова Ю.В.
подпись Ф.И.О.

Руководитель практики от ДВФУ _____Зотов С.С.
подпись Ф.И.О.

Оглавление

Задание на практику

- Практическое изучение алгоритмов кластеризации данных.

- Написание отчета по практике о проделанной работе.

Введение

Учебная практика (учебно-лабораторный практикум) проходила на кафедре информационной безопасности ШЕН ДВФУ в период с 22 февраля 2021 года по 26 июня 2021 года.

Целью прохождения практики является приобретение практических и теоретических навыков по специальности, а также навыков оформления проведенного исследования в отчетной форме.

Задачи практики:

1. Изучить существующие алгоритмы кластеризации данных.
2. Выбрать датасет и выполнить на нем несколько алгоритмов кластеризации.
3. Сравнить выбранные алгоритмы между собой.
4. На основе полученных знаний написать отчет по практике о проделанной работе.

1 Кластеризация

Кластерный анализ или кластеризация – это задача группировки набора объектов таким образом, чтобы объекты в одной группе (называемой кластером) были более похожи (в некотором смысле) друг на друга, чем на объекты в других группах (кластерах). Это основная задача исследовательского анализа данных и общий метод статистического анализа данных, используемый во многих областях, включая распознавание образов, анализ изображений, поиск информации, биоинформатику, сжатие данных, компьютерную графику и машинное обучение.

Применение кластерного анализа в общем виде сводится к следующим этапам:

1) Отбор выборки объектов для кластеризации.

2) Определение множества переменных, по которым будут оцениваться объекты в выборке. При необходимости – нормализация значений переменных.

3) Вычисление значений меры сходства между объектами.

4) Применение метода кластерного анализа для создания групп сходных объектов (кластеров).

5) Представление результатов анализа.

После получения и анализа результатов возможна корректировка выбранной метрики и метода кластеризации до получения оптимального результата.

Применение кластеризации несет в себе несколько целей:

- Понимание данных путём выявления кластерной структуры. Разбиение выборки на группы схожих объектов позволяет упростить дальнейшую

обработку данных и принятия решений, применяя к каждому кластеру свой метод анализа.

- Сжатие данных. Если исходная выборка избыточно большая, то можно сократить её, оставив по одному наиболее типичному представителю от каждого кластера.

- Обнаружение новизны. Выделяются нетипичные объекты, которые не удаётся присоединить ни к одному из кластеров.

Существует также метод группировки набора объектов, называемый классификация. Классификация — один из разделов машинного обучения, посвященный решению следующей задачи. Имеется множество объектов (ситуаций), разделённых, некоторым образом, на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется обучающей выборкой. Классовая принадлежность остальных объектов не известна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества.

Кластеризация отличается от классификации тем, что изначально не задано множество объектов, для которых известно, к каким классам они относятся, и даже могут быть неизвестны сами классы.

Решение задачи кластеризации принципиально неоднозначно, и тому есть несколько причин:

- Не существует однозначно наилучшего критерия качества кластеризации. Известен целый ряд эвристических критериев, а также ряд алгоритмов, не имеющих чётко выраженного критерия, но осуществляющих достаточно разумную кластеризацию «по построению». Все они могут давать разные результаты.

- Число кластеров, как правило, неизвестно заранее и устанавливается в соответствии с некоторым субъективным критерием.

- Результат кластеризации существенно зависит от метрики, выбор которой, как правило, также субъективен и определяется экспертом.

1.1 K-means кластеризация

Метод k-means — наиболее популярный метод кластеризации. Действие алгоритма таково, что он стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров:

$$V = \sum_{i=1}^{k} \sum_{x \in S_i} (x - \mu_i)^2$$

где k – число кластеров, S_i – полученные кластеры, $i = 1,2, \dots, k$, а μ_i – центры масс всех векторов x из кластера S_i.

Данный алгоритм разбивает множество элементов векторного пространства на заранее известное число кластеров *k*.

Основная идея заключается в том, что на каждой итерации перевычисляется центр масс для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике.

Алгоритм завершается, когда на какой-то итерации не происходит изменения внутрикластерного расстояния. Это происходит за конечное число итераций, так как количество возможных разбиений конечного множества конечно, а на каждом шаге суммарное квадратичное отклонение *V* уменьшается, поэтому зацикливание невозможно.

Проблемы k-means:

- Не гарантируется достижение глобального минимума суммарного квадратичного отклонения *V*, а только одного из локальных минимумов.
- Результат зависит от выбора исходных центров кластеров, их оптимальный выбор неизвестен.
- Число кластеров надо знать заранее.

1.2 Агломеративная кластеризация

Иерархическая кластеризация (также графовые алгоритмы кластеризации и иерархический кластерный анализ) — совокупность алгоритмов упорядочивания данных, направленных на создание иерархии (дерева) вложенных кластеров. Выделяют два класса методов иерархической кластеризации:

- Агломеративные методы: новые кластеры создаются путем объединения более мелких кластеров и, таким образом, дерево создается от листьев к стволу;

- Дивизивные или дивизионные методы: новые кластеры создаются путем деления более крупных кластеров на более мелкие и, таким образом, дерево создается от ствола к листьям. На практике такой подход нигде не применяется из-за больших вычислительных трудностей так как необходимо рассчитать большое количество всевозможных комбинаций деления.

Алгоритмы иерархической кластеризации предполагают, что анализируемое множество объектов характеризуется определённой степенью связности. Как и большинство визуальных способов представления зависимостей графы быстро теряют наглядность при увеличении числа кластеров.

Под дендрограммой обычно понимается дерево, построенное по матрице мер близости. Дендрограмма позволяет изобразить взаимные связи между объектами из заданного множества. Для создания дендрограммы требуется матрица сходства (или различия), которая определяет уровень сходства между парами кластеров.

2 Подготовка данных

Используется датасет UNSW_NB15 | Kaggle для классификации по обнаружению сетевых атак. Датасет содержит девять типов атак, а именно: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode и Worms, и 48 признаков.

Импортируем все нужные библиотеки и зададим параметры для лучшего отображения графиков:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
warnings.filterwarnings('ignore')#чтобы не светились всякие предупреждения
pd.set_option('display.max_columns', 999)#чтоб датасет был виден целиком
from sklearn.metrics import silhouette_score
from sklearn.metrics.cluster import homogeneity_score
```

```
from itertools import cycle, islice
plt.rcParams['figure.figsize'] = 12, 8#чтоб графики были большими и наглядными
import numpy as np
np.random.seed(0)#фиксируем инициализацию рандома чтобы у нас совпадали эксперименты
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
```

Рис. 1.Импорт всех нужных библиотек.

Читаем датасет и отображаем первые 5 строк:

```
data=pd.read_csv('UNSW_NB15_testing-set.csv')
data.head()
```

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload	dload	sloss	dloss	sinpkt	dinpkt
0	1	0.121478	tcp	-	FIN	6	4	258	172	74.087490	252	254	14158.942380	8495.365234	0	0	24.295600	8.375000
1	2	0.649902	tcp	-	FIN	14	38	734	42014	78.473372	62	252	8395.112305	503571.312500	2	17	49.915000	15.432865
2	3	1.623129	tcp	-	FIN	8	16	364	13186	14.170161	62	252	1572.271851	60929.230470	1	6	231.875571	102.737203
3	4	1.681642	tcp	ftp	FIN	12	12	628	770	13.677108	62	252	2740.178955	3358.622070	1	3	152.876547	90.235726
4	5	0.449454	tcp	-	FIN	10	6	534	268	33.373826	254	252	8561.499023	3987.059814	2	1	47.750333	75.659602

Рис. 2.Чтение датасета.

Получаем информацию о датафрейме:

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

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 175341 entries, 0 to 175340
        Data columns (total 45 columns):
```

```
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   id                 175341 non-null  int64
 1   dur                175341 non-null  float64
 2   proto              175341 non-null  object
 3   service            175341 non-null  object
 4   state              175341 non-null  object
 5   spkts              175341 non-null  int64
 6   dpkts              175341 non-null  int64
 7   sbytes             175341 non-null  int64
 8   dbytes             175341 non-null  int64
 9   rate               175341 non-null  float64
 10  sttl               175341 non-null  int64
 11  dttl               175341 non-null  int64
 12  sload              175341 non-null  float64
 13  dload              175341 non-null  float64
 14  sloss              175341 non-null  int64
 15  dloss              175341 non-null  int64
 16  sinpkt             175341 non-null  float64
 17  dinpkt             175341 non-null  float64
 18  sjit               175341 non-null  float64
 19  djit               175341 non-null  float64
 20  swin               175341 non-null  int64
 21  stcpb              175341 non-null  int64
 22  dtcpb              175341 non-null  int64
 23  dwin               175341 non-null  int64
 24  tcprtt             175341 non-null  float64
 25  synack             175341 non-null  float64
 26  ackdat             175341 non-null  float64
 27  smean              175341 non-null  int64
 28  dmean              175341 non-null  int64
 29  trans_depth        175341 non-null  int64
 30  response_body_len  175341 non-null  int64
 31  ct_srv_src         175341 non-null  int64
 32  ct_state_ttl       175341 non-null  int64
 33  ct_dst_ltm         175341 non-null  int64
 34  ct_src_dport_ltm   175341 non-null  int64
 35  ct_dst_sport_ltm   175341 non-null  int64
 36  ct_dst_src_ltm     175341 non-null  int64
 37  is_ftp_login       175341 non-null  int64
 38  ct_ftp_cmd         175341 non-null  int64
 39  ct_flw_http_mthd   175341 non-null  int64
 40  ct_src_ltm         175341 non-null  int64
 41  ct_srv_dst         175341 non-null  int64
 42  is_sm_ips_ports    175341 non-null  int64
 43  attack_cat         175341 non-null  object
dtypes: float64(11), int64(30), object(4)
memory usage: 60.2+ MB
```

Рис. 3. Получение информации о датафрейме.

Категорийные и числовые параметры:

```
cat_params=['proto','service','state','attack_cat']
num_params=['dur','spkts','dpkts','sbytes','dbytes','rate','sttl','dttl','sload','dload','sinpkt','dinpkt','sjit','djit',
            'swin','stcpb','dtcpb','dwin','tcprtt','synack','ackdat','smean','dmean','trans_depth','response_body_len',
            'ct_srv_src','ct_srv_src','ct_state_ttl','ct_dst_ltm','ct_src_dport_ltm','ct_dst_sport_ltm','ct_dst_src_ltm',
            'is_ftp_login','ct_ftp_cmd','ct_flw_http_mthd','ct_src_ltm','ct_srv_dst','is_sm_ips_ports']
```

Рис. 4.Категорийные и числовые параметры.

2.1 EDA

Вспомогательная функция, которая вычисляет все описательные все описательные статистические характеристики параметра такие как минимум, максимум, значения и строит график распределения:

```
def num_describer(df,param,bins=20):
    print('Описание параметра ',param)
    nulls=round(df[param].isnull().sum()/len(df[param]),4)
    low = df[param].min()
    median = df[param].median()
    mean = df[param].mean()
    hight = df[param].max()
    IQR = df[param].quantile(
        0.75) - df[param].quantile(0.25)
    perc25 = df[param].quantile(0.25)
    perc75 = df[param].quantile(0.75)

    print('доля пропусков : {} \n\r'.format(nulls),
          'min : {} \n\r'.format(low),
          '25-й перцентиль: {} \n\r'.format(perc25),
          'медиана: {} \n\r'.format(median),
          'среднее: {} \n\r'.format(mean),
          'max : {} \n\r'.format(hight),
          '75-й перцентиль: {} \n\r'.format(perc75),
          "IQR: {} \n\r".format(IQR),
          "Границы выбросов: [{f}, {l}] \n\r".format(f=perc25 - 1.5*IQR, l=perc75 + 1.5*IQR),
         )
    fig, (ax1, ax2) = plt.subplots(
    nrows=1, ncols=2,
    figsize=(8, 4))
    sns.distplot(df[param],ax=ax1)
    str1='Распределение значений '+param
    ax1.set_title(str1)
    ax1.legend()
    sns.distplot(np.log(df[param]+1),ax=ax2)
    str2='Логорифмированное Распределение значений '+param
    ax2.set_title(str2)
    ax2.legend()
    plt.show()
```

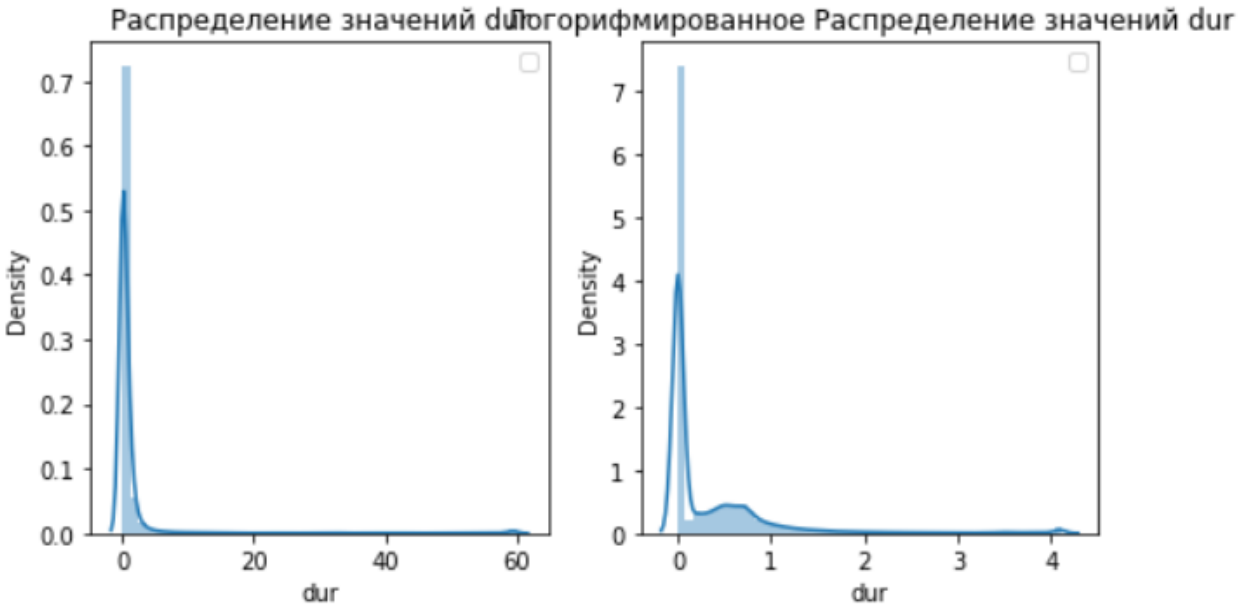
Рис. 4.Вспомогательная функция.

Сделаем разведку числовых параметров:

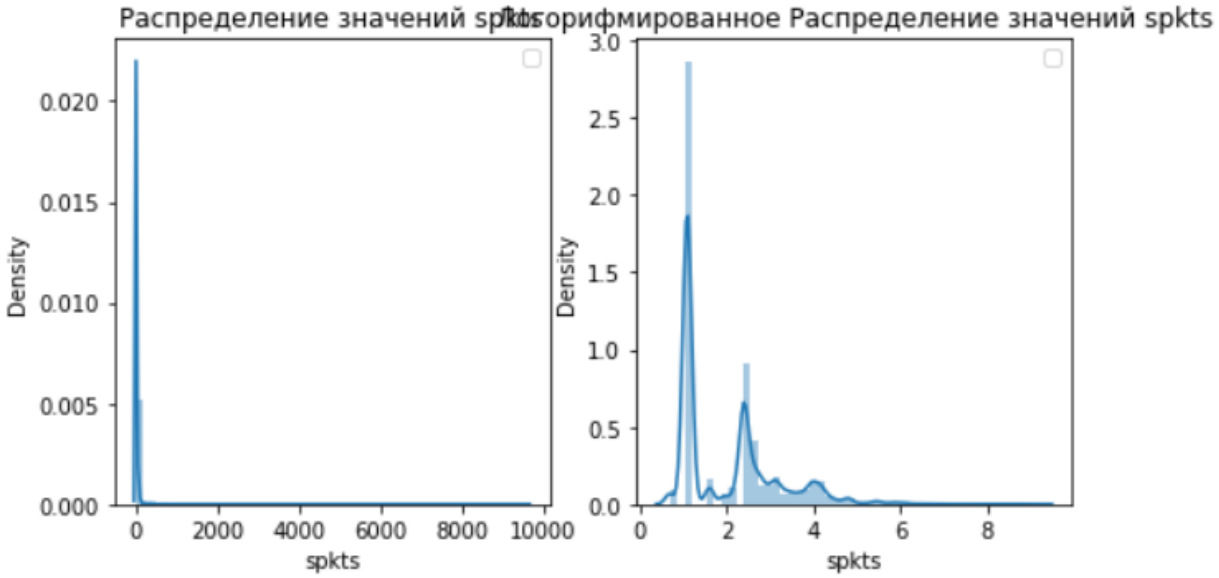
```
for i in num_params:
    num_describer(data,i)
```

Рис. 5.Разведка числовых параметров.

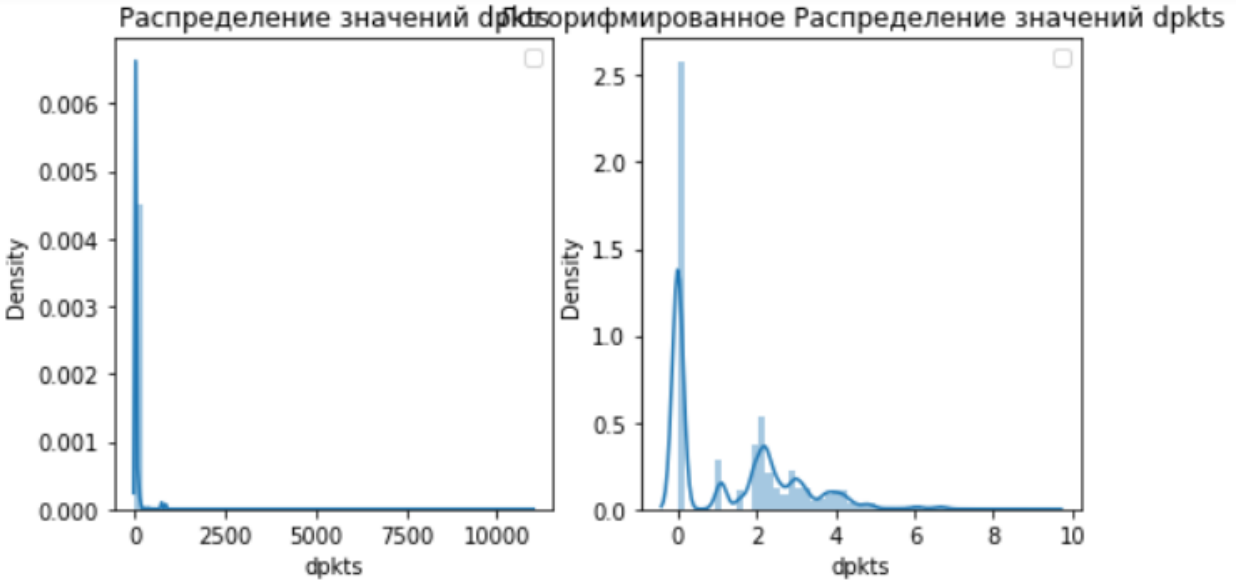
```
Описание параметра  dur
доля пропусков : 0.0
 min : 0.0
 25-й перцентиль: 8e-06
 медиана: 0.001582
 среднее: 1.3593886921261156
 max : 59.999989
 75-й перцентиль: 0.668069
 IQR: 0.668061
 Границы выбросов: [-1.0020835000000001, 1.6701605000000002]
```



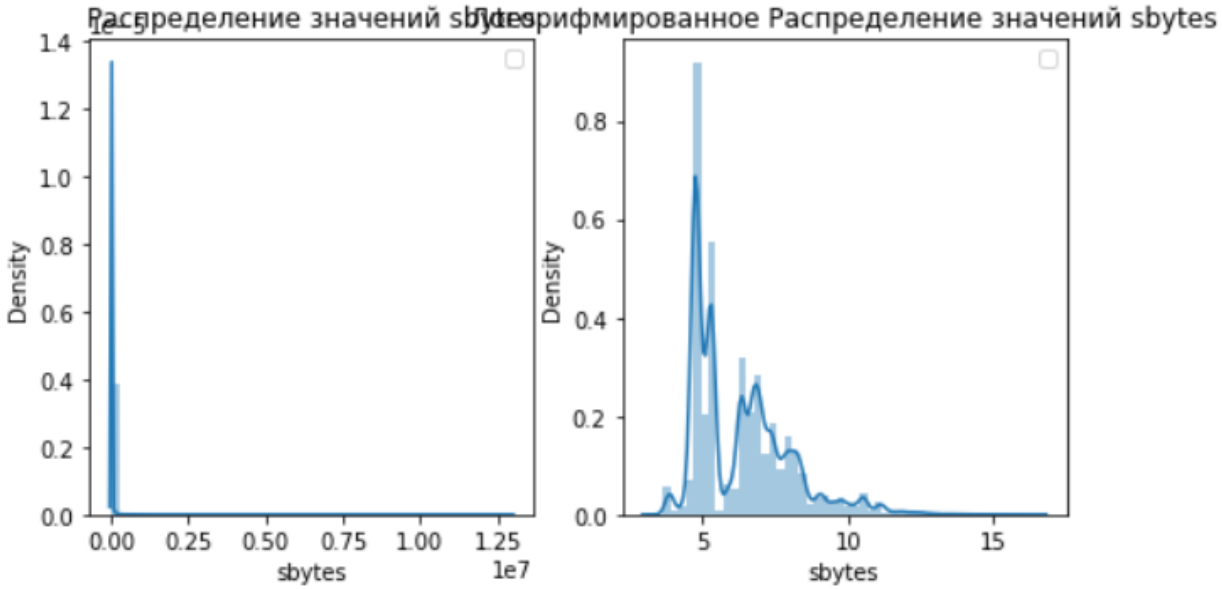

```
Описание параметра  spkts
доля пропусков : 0.0
 min : 1
 25-й перцентиль: 2.0
 медиана: 2.0
 среднее: 20.29866374664226
 max : 9616
 75-й перцентиль: 12.0
 IQR: 10.0
 Границы выбросов: [-13.0, 27.0]
```



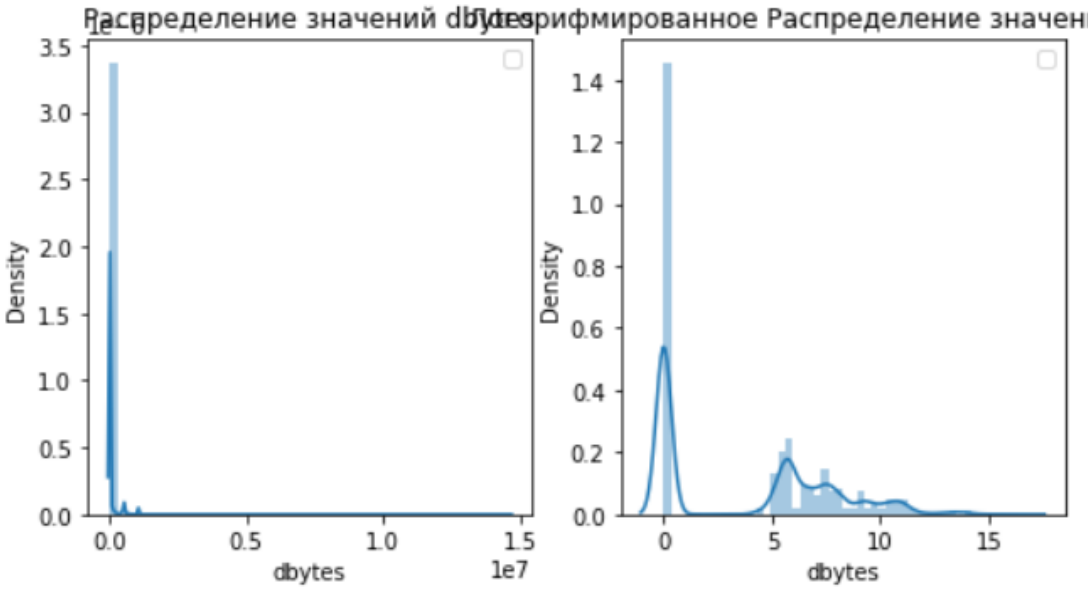

```
Описание параметра  dpkts
доля пропусков : 0.0
 min : 0
 25-й перцентиль: 0.0
 медиана: 2.0
 среднее: 18.969590683297117
 max : 10974
 75-й перцентиль: 10.0
 IQR: 10.0
 Границы выбросов: [-15.0, 25.0]
```



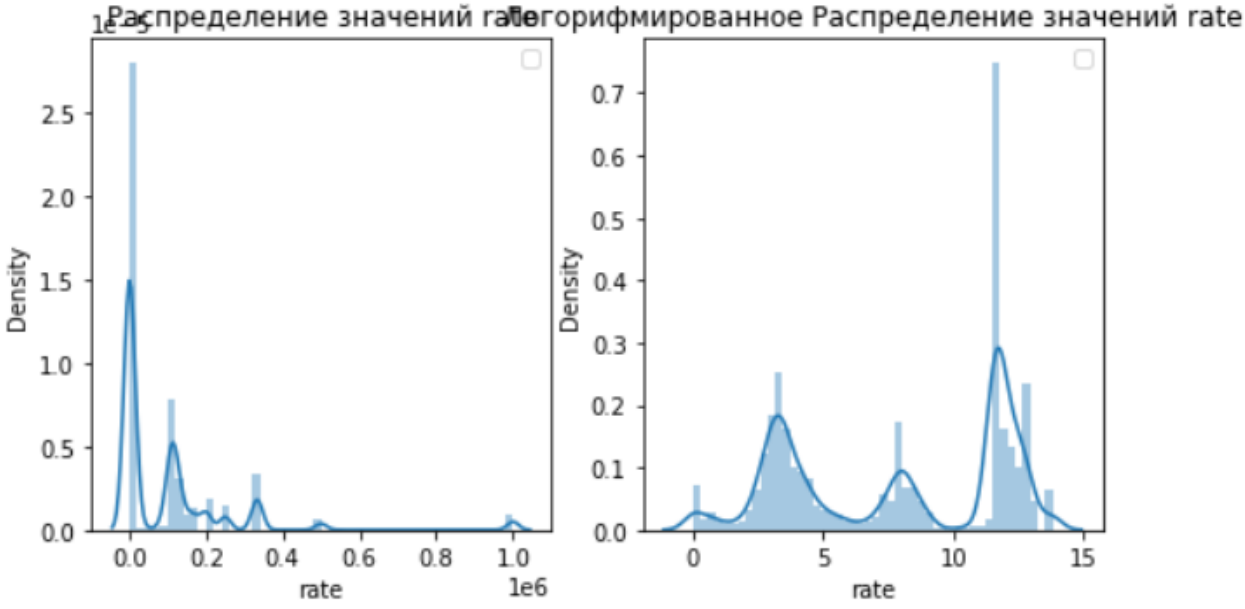
```
Описание параметра  sbytes
доля пропусков : 0.0
 min : 28
 25-й перцентиль: 114.0
 медиана: 430.0
 среднее: 8844.843835725815
 max : 12965233
 75-й перцентиль: 1418.0
 IQR: 1304.0
 Границы выбросов: [-1842.0, 3374.0]
```



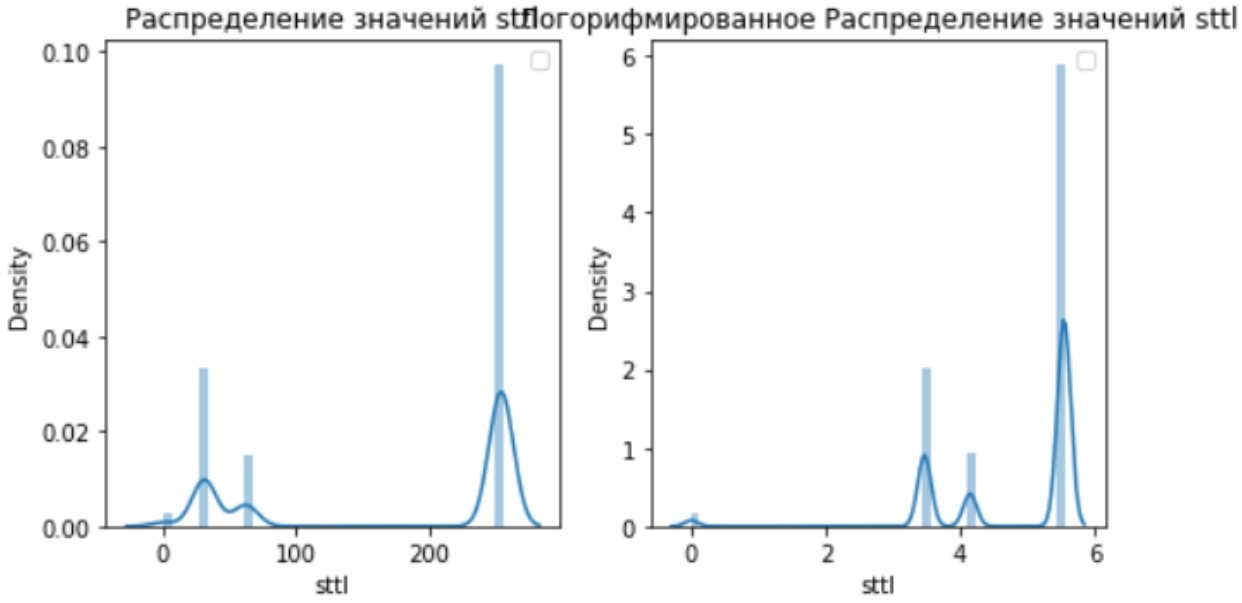

```
Описание параметра  dbytes
доля пропусков : 0.0
 min : 0
 25-й перцентиль: 0.0
 медиана: 164.0
 среднее: 14928.918564397374
 max : 14655550
 75-й перцентиль: 1102.0
 IQR: 1102.0
 Границы выбросов: [-1653.0, 2755.0]
```



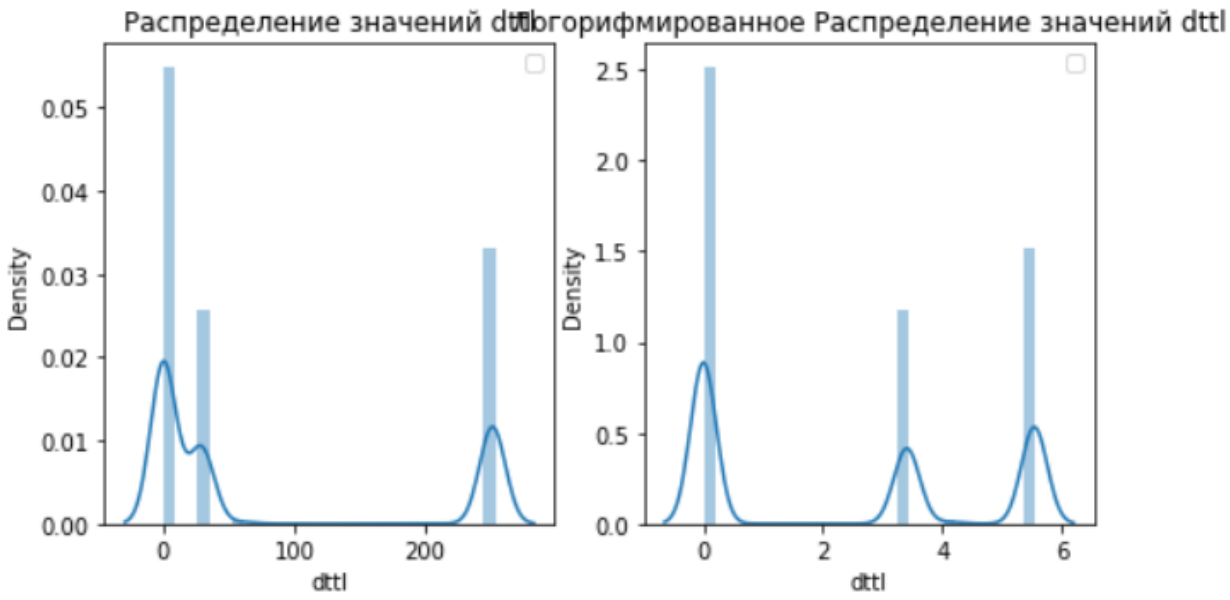

```
Описание параметра  rate
доля пропусков : 0.0
 min : 0.0
 25-й перцентиль: 32.78614
 медиана: 3225.80652
 среднее: 95406.18710525581
 max : 1000000.003
 75-й перцентиль: 125000.0003
 IQR: 124967.21416
 Границы выбросов: [-187418.0351, 312450.82154000003]
```



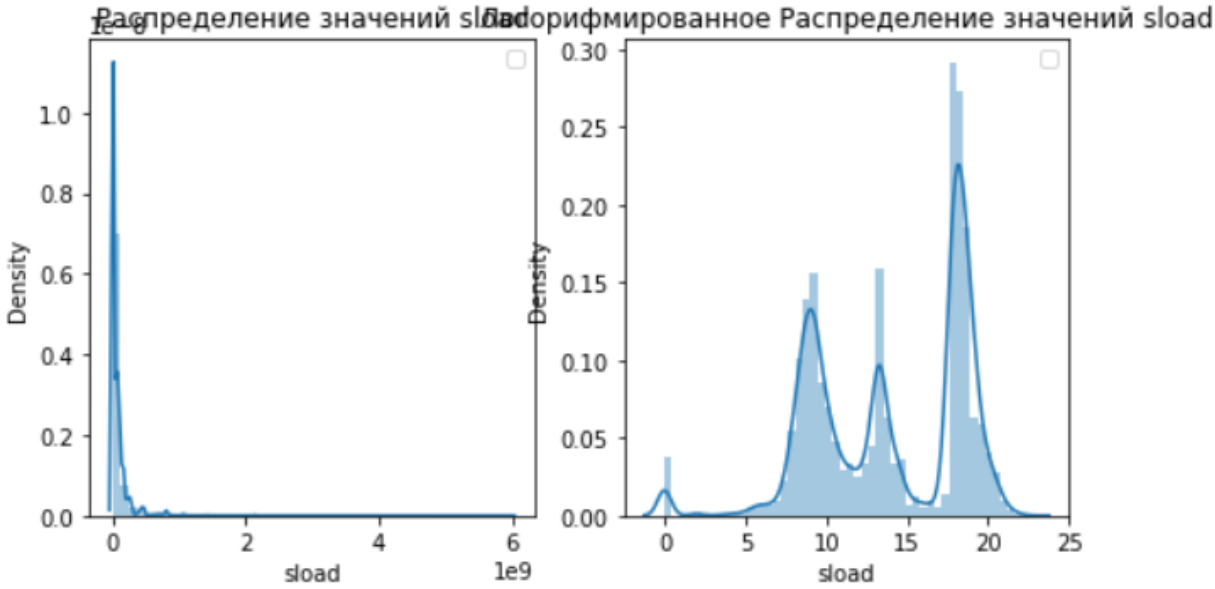
```
Описание параметра  sttl
доля пропусков : 0.0
 min : 0
 25-й перцентиль: 62.0
 медиана: 254.0
 среднее: 179.546996994428
 max : 255
 75-й перцентиль: 254.0
 IQR: 192.0
 Границы выбросов: [-226.0, 542.0]
```




```
Описание параметра  dttl
доля пропусков : 0.0
 min : 0
 25-й перцентиль: 0.0
 медиана: 29.0
 среднее: 79.60956650184498
 max : 254
 75-й перцентиль: 252.0
 IQR: 252.0
 Границы выбросов: [-378.0, 630.0]
```




```
Описание параметра  sload
доля пропусков : 0.0
 min : 0.0
 25-й перцентиль: 13053.33887
 медиана: 879674.75
 среднее: 73454033.19406345
 max : 5988000256.0
 75-й перцентиль: 88888888.0
 IQR: 88875834.66113
 Границы выбросов: [-133300698.65282498, 222202639.991695]
```

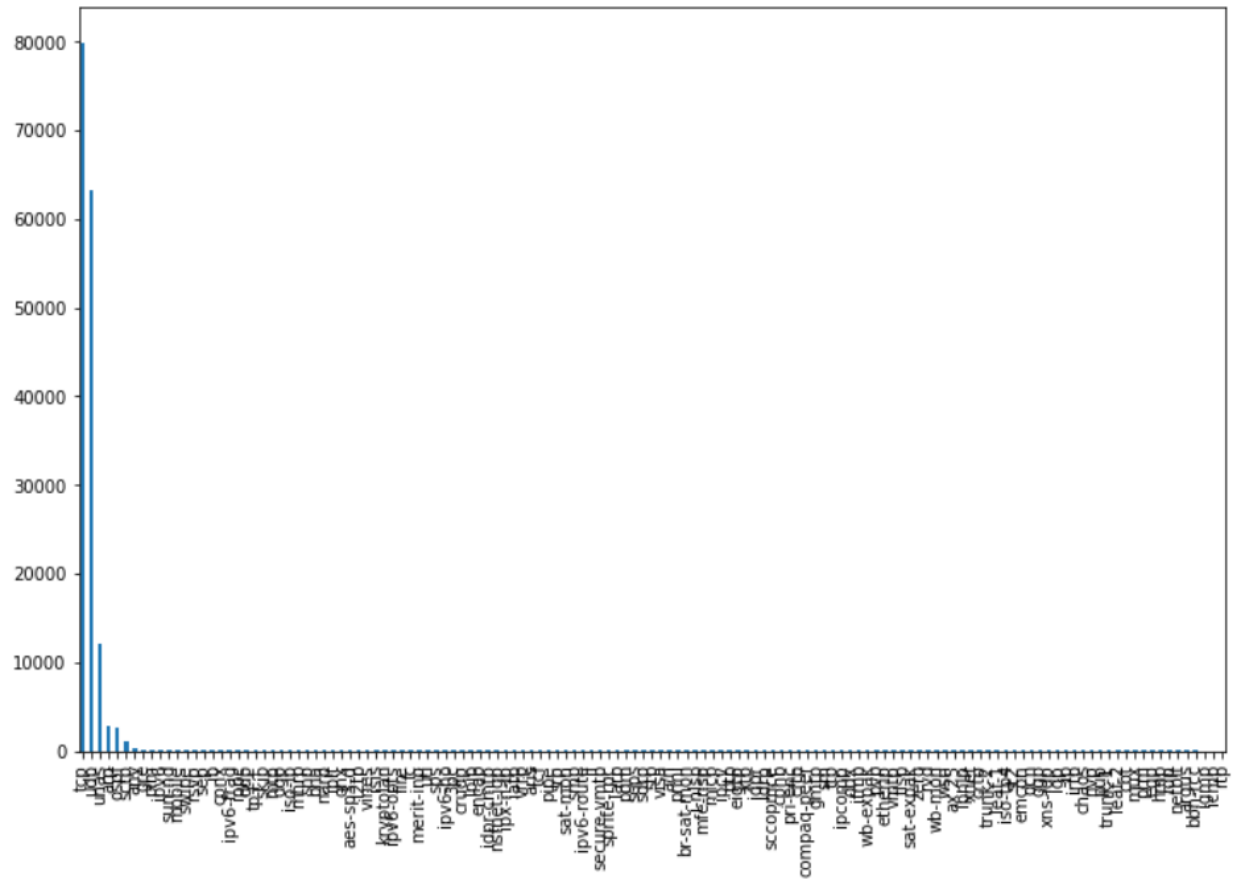


```
Описание параметра  dload
доля пропусков : 0.0
 min : 0.0
 25-й перцентиль: 0.0
 медиана: 1447.022705
 среднее: 671205.5741882676
 max : 22422730.0
 75-й перцентиль: 27844.87109
 IQR: 27844.87109
 Границы выбросов: [-41767.306635, 69612.177725]
```

Сделаем осмотр категорийных параметров:

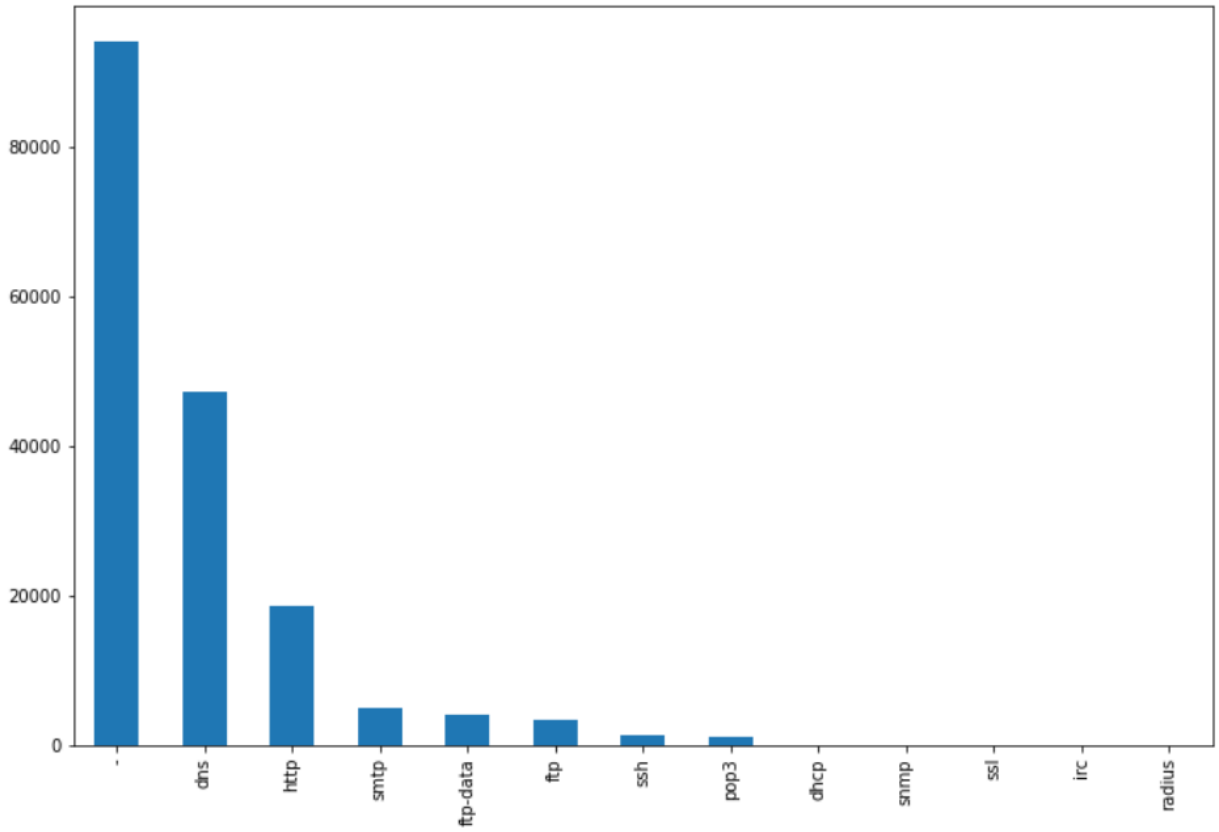
```
obj_param_describe(data,'proto')
```

```
доля пропусков:  0.0
Значения параметра и их количество
tcp        79946
udp        63283
unas       12084
arp         2859
ospf        2595
           ...
argus         98
bbn-rcc       98
igmp          18
icmp          15
rtp            1
Name: proto, Length: 133, dtype: int64
```



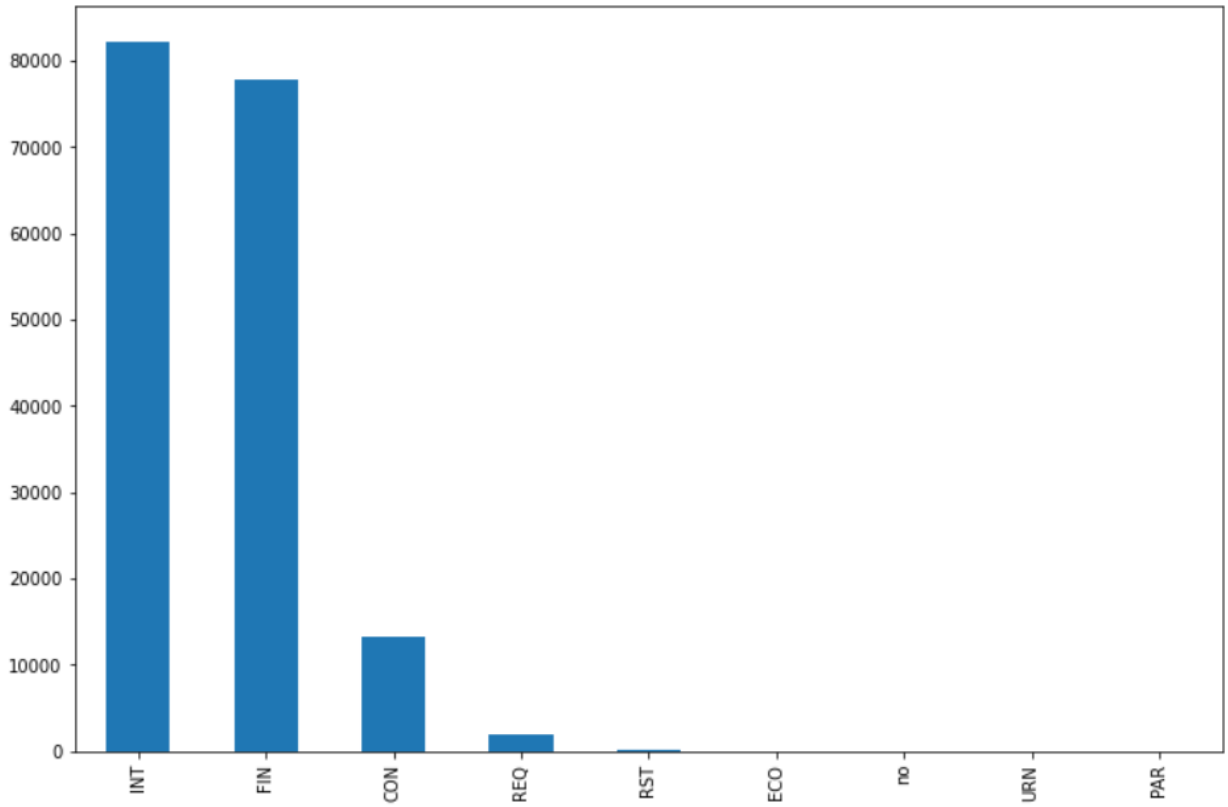

```
obj_param_describe(data,'service')
```

```
доля пропусков:  0.0
Значения параметра и их количество
-           94168
dns         47294
http        18724
smtp         5058
ftp-data     3995
ftp          3428
ssh          1302
pop3         1105
dhcp           94
snmp           80
ssl            56
irc            25
radius         12
Name: service, dtype: int64
```




```
obj_param_describe(data,'state')
```

```
доля пропусков:  0.0
Значения параметра и их количество
INT    82275
FIN    77825
CON    13152
REQ     1991
RST       83
ECO       12
no         1
URN        1
PAR        1
Name: state, dtype: int64
```



Типы атак, которые и будут нашими классами, на которые мы будем кластеризовать:

```
obj_param_describe(data,'attack_cat')
```

```
доля пропусков:  0.0
Значения параметра и их количество
Normal            56000
Generic           40000
Exploits          33393
Fuzzers           18184
DoS               12264
Reconnaissance    10491
Analysis           2000
Backdoor           1746
Shellcode          1133
Worms               130
Name: attack_cat, dtype: int64
```

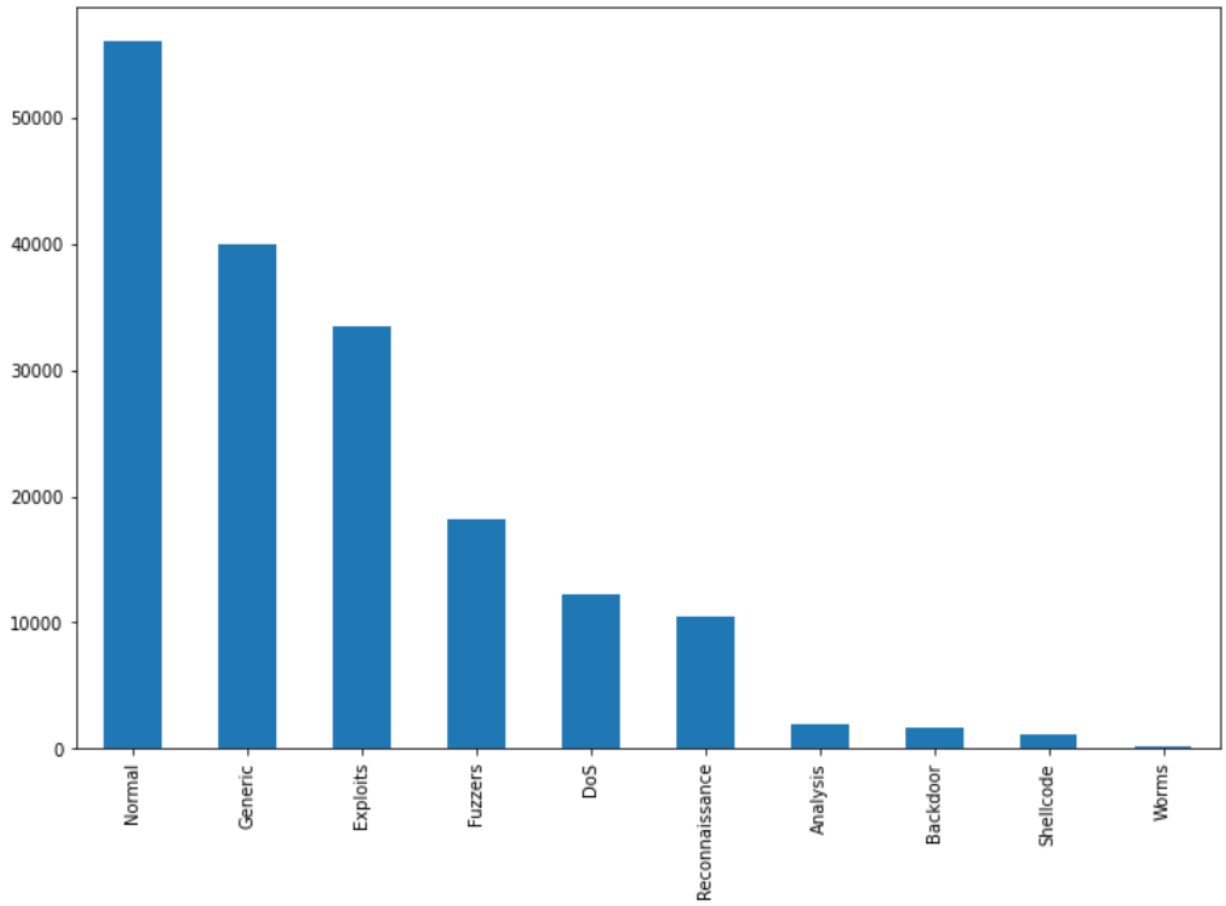


Рис. 5. Осмотр категорийных параметров.

2.2 Подготовка датасета

```
data_res=data.drop(['id','state','service','proto','label'],axis=1)#удаляем ненужные столбцы
```

Рис. 6.Удаление ненужных столбцов.

Неполный датафрейм:

	dur	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload	dload	sloss	dloss	sinpkt	dinpkt	sjit
0	0.121478	6	4	258	172	74.087490	252	254	1.415894e+04	8495.365234	0	0	24.295600	8.375000	30.177547
1	0.649902	14	38	734	42014	78.473372	62	252	8.395112e+03	503571.312500	2	17	49.915000	15.432865	61.426934
2	1.623129	8	16	364	13186	14.170161	62	252	1.572272e+03	60929.230470	1	6	231.875571	102.737203	17179.586860
3	1.681642	12	12	628	770	13.677108	62	252	2.740179e+03	3358.622070	1	3	152.876547	90.235726	259.080172
4	0.449454	10	6	534	268	33.373826	254	252	8.561499e+03	3987.059814	2	1	47.750333	75.659602	2415.837634
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
175336	0.000009	2	0	114	0	111111.107200	254	0	5.066666e+07	0.000000	0	0	0.009000	0.000000	0.000000
175337	0.505762	10	8	620	354	33.612649	254	252	8.826286e+03	4903.492188	2	1	54.400111	66.980570	3721.068786
175338	0.000009	2	0	114	0	111111.107200	254	0	5.066666e+07	0.000000	0	0	0.009000	0.000000	0.000000
175339	0.000009	2	0	114	0	111111.107200	254	0	5.066666e+07	0.000000	0	0	0.009000	0.000000	0.000000
175340	0.000009	2	0	114	0	111111.107200	254	0	5.066666e+07	0.000000	0	0	0.009000	0.000000	0.000000

175341 rows × 40 columns

Рис. 7.Неполный датафрейм.

Алгоритмы работают с матрицами или векторами, поэтому подать датафрейм на вход не получится, берем только значения:

```
X_data=data_res.drop(['attack_cat'],axis=1).values
Y_data=data_res.attack_cat.values
X_data
```

```
array([[1.214780e-01, 6.000000e+00, 4.000000e+00, ..., 1.000000e+00,
        1.000000e+00, 0.000000e+00],
       [6.499020e-01, 1.400000e+01, 3.800000e+01, ..., 1.000000e+00,
        6.000000e+00, 0.000000e+00],
       [1.623129e+00, 8.000000e+00, 1.600000e+01, ..., 2.000000e+00,
        6.000000e+00, 0.000000e+00],
       ...,
       [9.000000e-06, 2.000000e+00, 0.000000e+00, ..., 3.000000e+00,
        1.200000e+01, 0.000000e+00],
       [9.000000e-06, 2.000000e+00, 0.000000e+00, ..., 3.000000e+01,
        3.000000e+01, 0.000000e+00],
       [9.000000e-06, 2.000000e+00, 0.000000e+00, ..., 3.000000e+01,
        3.000000e+01, 0.000000e+00]])
```

```
Y_data
```

```
array(['Normal', 'Normal', 'Normal', ..., 'Generic', 'Generic', 'Generic'],
      dtype=object)
```

3 Демонстрация работы алгоритма

3.1 Алгоритм k-means

Выберем оптимальное кол-во кластеров для наших данных: в нашем случае это 10 кластеров(по виду атак),один из которых в центре.

Добавим случайным образом 10 центроидов. Центроиды – это предполагаемые центры будущих кластеров. Для каждой точки мы можем посчитать, к какому центроиду она ближе и окрасить в соответствующий цвет. Перенесем центроид в центр выборки, к которой мы его отнесли: то есть расположим его так, чтобы расстояния от объектов кластера до центроида были как можно меньше. Повторим эти шаги, пока алгоритм не сойдется.

```
from sklearn.cluster import KMeans

k_means = KMeans(n_clusters=10,  #количество кластеров
                 init='k-means++', # 'k-means++', 'random', numpy.array метод инициализации
                 max_iter=1000 #максимальное количество итераций
                )
k means.fit(X data)#запускаем кластеризацию
```

```
KMeans(max_iter=1000, n_clusters=10)
```

Выведем лэйблы, или метки кластера, которые выдал наш алгоритм для обучающей выборки.

```
k_means.labels_
```

```
array([8, 3, 5, ..., 0, 0, 0])
```

Предсказания в виде вектора не особо информативны. Приклеим наши предсказания к датасету data_res. Если отлистать датасет в конец и посмотреть на столбцы attack_cat, cluster_label_kmeans видно, что много ошибок даже по первым строкам где класс Normal. Основные причины:

1.Большая размерность пространства.

2.Алгоритмы кластеризации не могут обрабатывать категорийные параметры, а если их включить то только усугубим проблему 1.

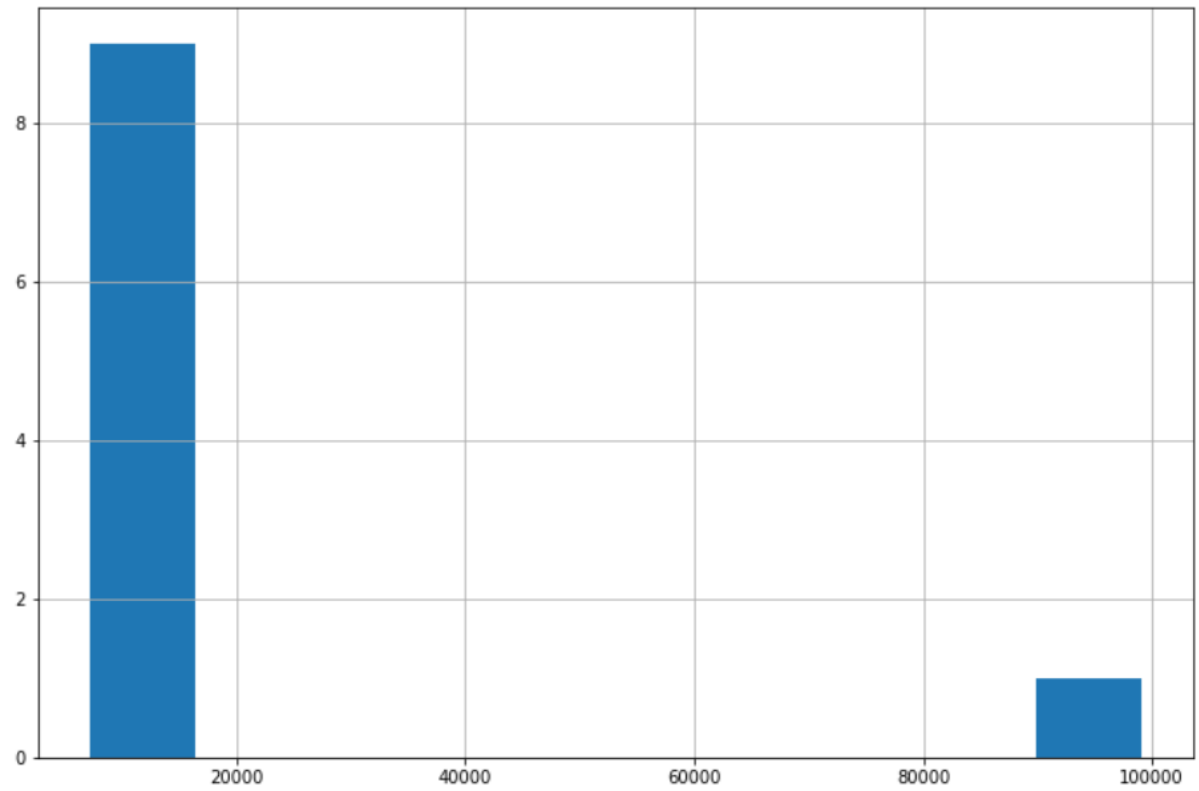
```
data_res['cluster_label_kmeans']=pd.DataFrame(k_means.labels_)
data_res
```

dport_ltm	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp_cmd	ct_flw_http_mthd	ct_src_ltm	ct_srv_dst	is_sm_ips_ports	attack_cat	cluster_label_kmeans
1	1	1	0	0	0	1	1	0	Normal	8
1	1	2	0	0	0	1	6	0	Normal	3
1	1	3	0	0	0	2	6	0	Normal	5
1	1	3	1	1	0	2	1	0	Normal	7
2	1	40	0	0	0	2	39	0	Normal	1
...	...	...	...	...	...	...	...	...	...	...
24	13	24	0	0	0	24	24	0	Generic	0
1	1	2	0	0	0	1	1	0	Shellcode	9
3	3	13	0	0	0	3	12	0	Generic	0
30	14	30	0	0	0	30	30	0	Generic	0
30	16	30	0	0	0	30	30	0	Generic	0

```
0    99032
3    10093
7     9582
8     9052
1     8910
5     8245
9     7960
4     7946
2     7290
6     7231
Name: cluster_label_kmeans, dtype: int64
```

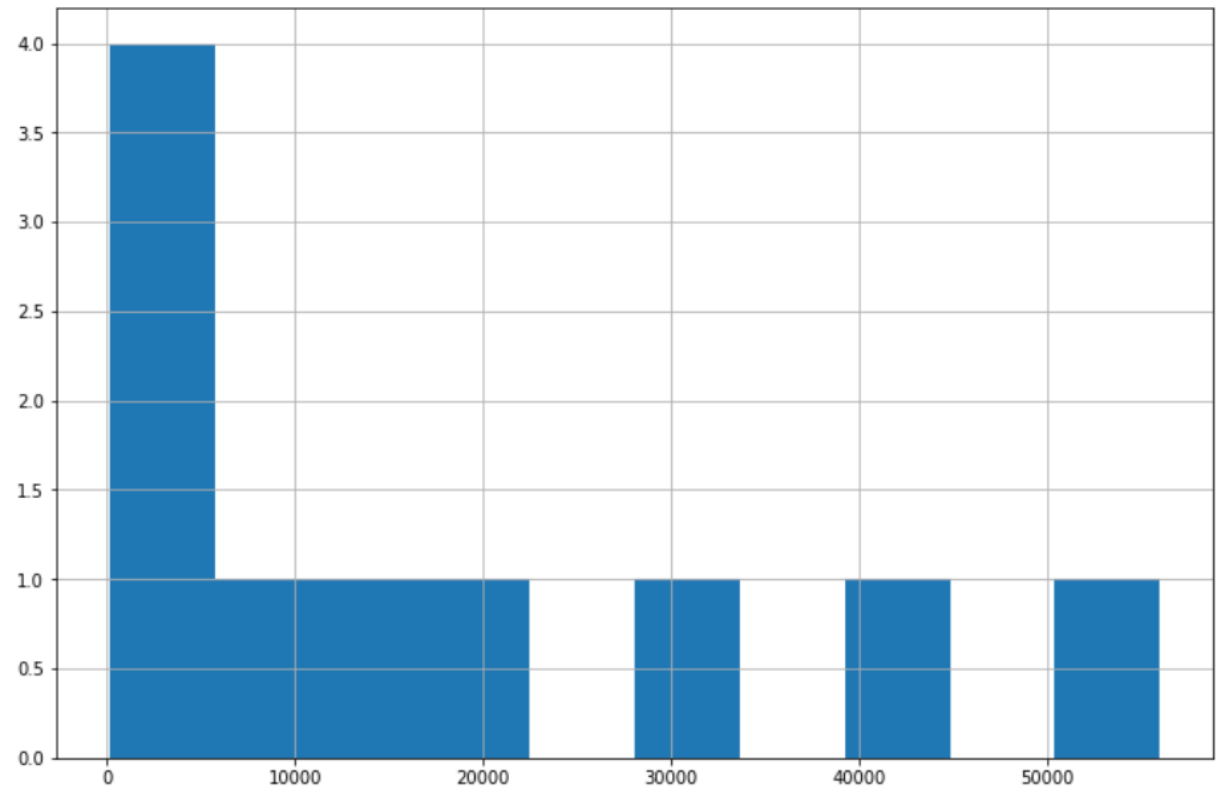
```
data_res['cluster_label_kmeans'].value_counts().hist()
```

```
<AxesSubplot:>
```




```
data_res['attack_cat'].value_counts().hist()
```

```
<AxesSubplot:>
```



Если сравнить две гистограммы выше, видно, что пропорция классов/кластеров нарушилась.

Посмотрим на координаты центров кластеров для каждого кластера. Из-за большой размерности получается не особо информативно.

```
k_means.cluster_centers_
```

```
array([[ 1.25686241e+00,  5.13647011e+00,  2.16226777e+00,
         1.29850253e+03,  1.63096036e+03,  1.68245938e+05,
         2.15843387e+02,  7.67406099e+00,  1.29699298e+08,
         1.31971222e+05,  3.64186187e-01,  6.86783118e-01,
         1.64978470e+03,  2.91460754e+01,  3.52760707e+02,
         4.54974178e+01,  9.38000808e+00,  8.00299844e+06,
         7.87006589e+06,  7.19376010e+00,  1.87208932e-03,
         9.48858885e-04,  9.23230432e-04,  1.06232441e+02,
         2.04328655e+01,  8.34006462e-03,  2.82468811e+02,
         1.35179120e+01,  1.79023627e+00,  9.04285137e+00,
         8.45166599e+00,  6.63204766e+00,  1.33844911e+01,
         9.08723748e-04,  9.08723748e-04,  7.13852989e-03,
         9.81747779e+00,  1.34445981e+01,  2.78675283e-02],
       [ 1.36343620e+00,  3.69735129e+01,  4.26615039e+01,
         1.40662525e+04,  3.51171044e+04,  8.88491885e+02,
         1.32498316e+02,  1.72238384e+02,  4.45947580e+05,
         1.43850776e+06,  9.16217733e+00,  1.62722783e+01,
         1.24474266e+02,  1.68030804e+02,  1.10791200e+04,
         1.35773621e+03,  2.54976431e+02,  1.95707881e+09,
```

После того как провели кластеризацию и известны центры кластеров для новых объектов, можно сразу считать к какому кластеру он относится.

Пример на первой строке датасета:

```
answer=k_means.predict(np.array([X_data[0]]))
answer[0]
```

```
8
```

3.2 Алгоритм агломеративной кластеризации

Как было сказано выше, иерархическая кластеризация бывает двух видов: агломеративная и дивизивная. Будем использовать агломеративную кластеризацию, так как дивизивная обладает большой вычислительной сложностью.

Создадим объект агломеративной кластеризации со следующими свойствами: разбиение ровно на 10 кластеров, использование евклидовой метрики связи, как самой простой (можно также использовать метрики L1, L2, манхэттенскую метрику, косинусную или вычисленную заранее вместе с

матрицей расстояний), использование критерия связи минимальной дисперсии Уорда (можно также использовать полный критерий, средний или одиночный). Код подготовки алгоритма представлен на рис. 1.

```
from sklearn.cluster import AgglomerativeClustering

ac = AgglomerativeClustering(n_clusters=10,
                             affinity='euclidean', # “euclidean”, “l1”, “l2”, “manhattan”,
                                                   # “cosine”, or “precomputed”
                             linkage='ward', # “ward”, “complete”, “average”, “single”
                            )
```

Рис. 8. Подготовка агломеративной кластеризации.

Затем кластеризуем матрицу X и посмотрим номера классов элементов, как указано на рис. 2.

```
ac.fit(X)# кластеризуем
ac.labels_
```

```
array([0, 5, 9, ..., 0, 7, 1], dtype=int64)
```

Рис. 9. Кластеризация.

Затем выведем кол-во элементов в кластерах, как указано на рис. 3. Таким образом алгоритм агломеративной кластеризации был выполним.

```
a = ac.labels_
_, counts = np.unique(a, return_counts=True)
for count in counts:
    print(count, end=' ')
```

```
226 184 171 136 103 183 138 137 103 119
```

Рис. 10. Количество элементов в кластерах.

Заключение

Для достижения данной цели, в процессе прохождения учебной практики (учебно-лабораторного практикума) изучила существующие методы кластеризации, а также сравнила их. На основе полученных знаний произвела кластеризацию выбранного датасета.

Также были изучены требования к написанию отчета по практике. В результате прохождения практики был составлен отчет по практике, соответствующий предъявленным требованиям.

В ходе прохождения практики все задачи были выполнены, а цель достигнута.