МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В. И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ
по ИДЗ
по дисциплине «Машинное обучение»

Студент гр. 1310 _____ Комаров Д. Е.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2025

Постановка задачи

Датасет «RT-IoT2022» содержит информацию о сетевом трафике и видах сетевых атак.

Задача заключается в предсказании типа сетевой атаки по признакам трафика сети (Задача классификации).

Описание используемого набора данных

Датасет содержит 123117 записей о сетевых атаках. Имеется 12 классов сетевых атак, представленных в таблице 1. Заметим, что классы имеют сильный дисбаланс.

Таблица 1 – Классы сетевых атак

Класс	Количество записей
MQTT_Publish	4146
Thing_Speak	8108
Wipro_bulb	253
ARP_poisioning	7750
DDOS_Slowloris	534
DOS_SYN_Hping	94659
Metasploit_Brute_Force_SSH	37
NMAP_FIN_SCAN	28
NMAP_OS_DETECTION	2000
NMAP_TCP_scan	1002
NMAP_UDP_SCAN	2590
NMAP_XMAS_TREE_SCAN	2010

Датасет имеет 83 признака. Признаки имеются категориальные, целочисленные, непрерывные. Категориальные признаки представлены в таблице 2.

Таблица 2 – Категориальные признаки

Имя	Значения
proto	*'udp', 'icmp', 'tcp'*

Продолжение таблицы 2

service	*'dns', '-', 'ntp', 'dhcp', 'ssh', 'irc', 'ssl', 'radius', 'http', 'mqtt'*

Целочисленные признаки представлены в таблице 3

Таблица 3 – Целочисленные признаки

Имя	Минимальное значение	Максимальное значение	Среднее значение	СКО
id.orig_p	0	65535	34639.26	19070.62
id.resp_p	0	65389	1014.31	5256.37
fwd_pkts_tot	0	4345	2.27	22.34
bwd_pkts_tot	0	10112	1.91	33.02
fwd_data_pkts_tot	0	4345	1.47	19.64
bwd_data_pkts_tot	0	10105	0.82	32.29
fwd_header_size_tot	0	69296	53.89	393.03
fwd_header_size_min	0	44	19.78	5.35
fwd_header_size_max	0	52	20.65	7.23
bwd_header_size_tot	0	323592	46.63	1028.23
bwd_header_size_min	0	40	17.7	8
bwd_header_size_max	0	44	18.43	9.41
flow_FIN_flag_count	0	10	0.12	0.48
flow_SYN_flag_count	0	8	0.95	0.47
flow_RST_flag_count	0	10	0.8	0.44
fwd_PSH_flag_count	0	864	0.35	3.95
bwd_PSH_flag_count	0	1446	0.39	6.01
flow_ACK_flag_count	0	11772	2.68	41.65
fwd_URG_flag_count	0	1	0.02	0.13
bwd_URG_flag_count	0	0	0	0
flow_CWR_flag_count	0	4	0.001	0.04

Продолжение таблицы 3

flow_ECE_flag_count	0	4	0.001	0.03
fwd_init_window_size	0	65535	6118.91	18716.31
bwd_init_window_size	0	65535	2739.78	10018.85
fwd_last_window_size	0	65535	751.65	6310.18

Гистограммы целочисленных признаков представлены на рисунке 1.

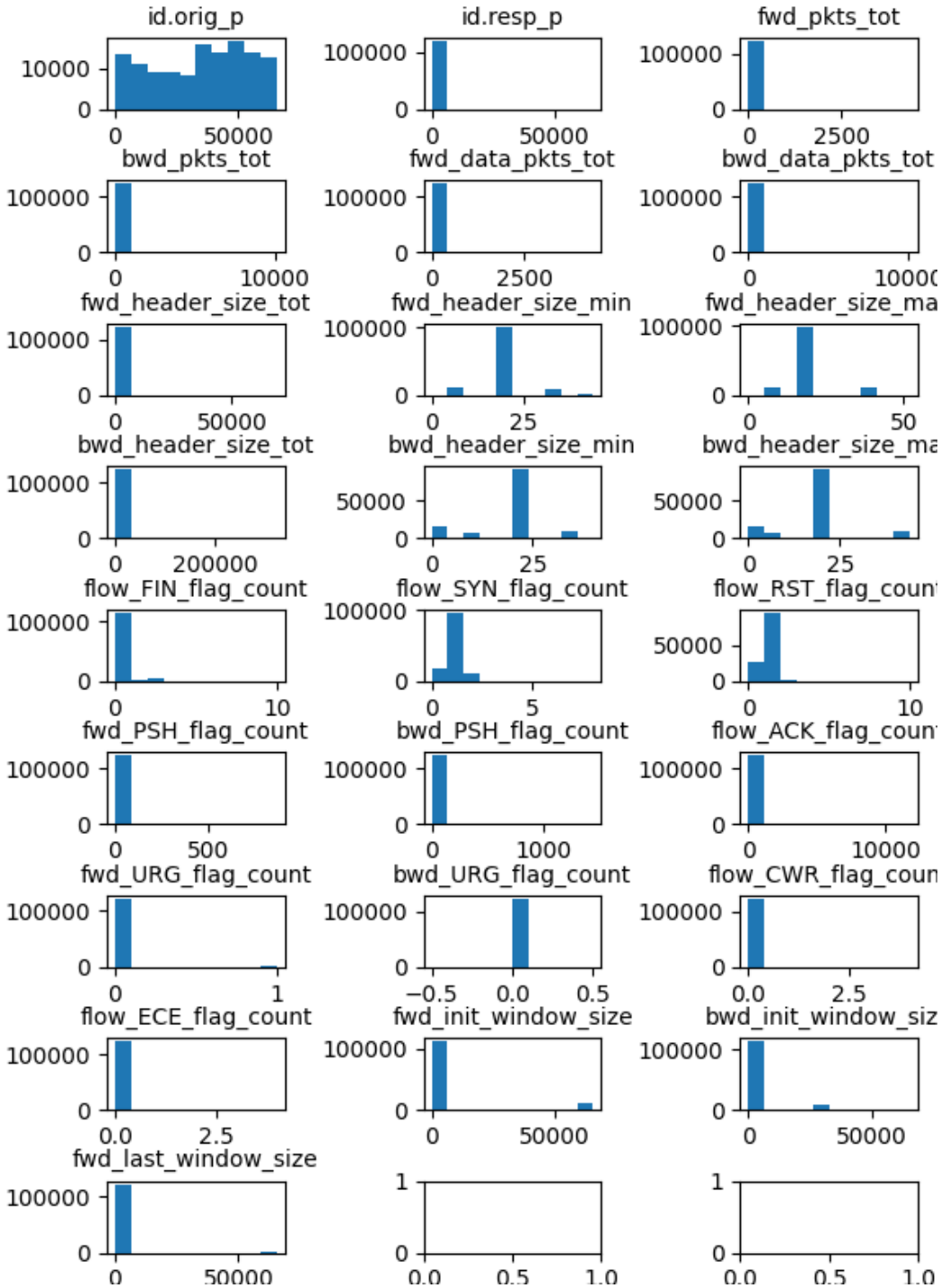

Рисунок 1 – Гистограммы целочисленных признаков

Непрерывные признаки представлены в таблице 4.

Таблица 4 – Непрерывные признаки

Имя	Минимальное значение	Максимальное значение	Среднее значение	СКО
flow_duration	0	21728.34	3.81	130.01
fwd_pkts_per_sec	0	1.05e+06	3.52e+05	3.715e+05
bwd_pkts_per_sec	0	1.05e+06	3.52e+05	3.715e+05
flow_pkts_per_sec	0	2.1e+06	7.04e+05	7.42e+05
down_up_ratio	0	6.088	0.85	0.34
fwd_pkts_payload.min	0	1097	96.26	45.27
fwd_pkts_payload.max	0	1420	120.75	121.3
fwd_pkts_payload.tot	0	747340	221.52	4820.4
fwd_pkts_payload.avg	0	1319.37	100.52	46.1
fwd_pkts_payload.std	0	731.58	8.11	45.04
bwd_pkts_payload.min	0	1357	3.82	19.61
bwd_pkts_payload.max	0	5124	52.41	231.31
bwd_pkts_payload.tot	0	1.36e+07	5.13e+02	4.24e+04
bwd_pkts_payload.avg	0	1457.05	18.79	83.48
bwd_pkts_payload.std	0	1506.01	20.55	93.4
flow_pkts_payload.min	0	1097	13.55	35.47
flow_pkts_payload.max	0	5124	148.51	218.23
flow_pkts_payload.tot	0	1.36e+07	7.35e+02	4.3e+04
flow_pkts_payload.avg	0	1156.08	65.01	50.41
flow_pkts_payload.std	0	924.65	76.04	74.02
fwd_iat.min	0	3e+08	8.84e+03	1.23e+06
fwd_iat.max	0	3e+08	1.72e+06	9.23e+06
fwd_iat.tot	0	2.17e+10	3.78e+06	1.3e+08
fwd_iat.avg	0	3e+08	2.37e+05	1.9e+06
fwd_iat.std	0	2.12e+08	5.78e+05	3.2e+06

Продолжение таблицы 4

bwd_iat.min	0	4.32e+07	3.76e+03	2.26e+05
bwd_iat.max	0	3e+08	4.08e+05	4.29e+06
bwd_iat.tot	0	1.88e+10	1.78e+06	9.12e+07
bwd_iat.avg	0	1.5e+08	8.77e+04	1.1e+06
bwd_iat.std	0	2.12e+08	1.47e+05	1.78e+06
flow_iat.min	0	4.35e+07	4.28e+03	2.54e+05
flow_iat.max	0	3e+08	1.73e+06	9.25e+06
flow_iat.tot	0	2.17e+10	3.81e+06	1.3e+08
flow_iat.avg	0	7.28e+07	1.4e+05	8.756e+05
flow_iat.std	0	1.34e+08	4.5e+05	2.51e+06
payload_bytes_per_second	0	1.26e+08	4.1e+07	4.496e+07
fwd_subflow_pkts	0	276.83	1.55	2.72
bwd_subflow_pkts	0	1685.33	1.33	6.01
fwd_subflow_bytes	0	52067.75	136.48	428.91
bwd_subflow_byte	0	2.27e+06	2.18e+02	7.59e+03
fwd_bulk_bytes	0	465095	19.25	1974.62
bwd_bulk_bytes	0	6.81e+06	1.55e+02	1.96e+04
fwd_bulk_packets	0	343	0.02	1.48
bwd_bulk_packets	0	5052.5	0.13	14.55
fwd_bulk_rate	0	4.63e+07	3.846e+03	3.08e+05
bwd_bulk_rate	0	2.83e+07	4.84e+04	6.85e+05
active.min	0	3.13e+08	1.33e+05	1.04e+06
active.max	0	8.48e+08	1.79e+05	3.01e+06
active.tot	0	2.95e+09	2.93e+05	1.45e+07
active.avg	0	4.37e+08	1.48e+05	1.61e+06
active.std	0	4.77e+08	2.35e+04	1.48e+06
idle.min	0	3e+08	1.62e+06	8.81e+06

Продолжение таблицы 4

idle.max	0	3e+08	1.7e+06	9.25e+06
idle.tot	0	2.1e+10	3.52e+06	1.23e+08
idle.avg	0	3e+08	1.66e+06	9.01e+06
idle.std	0	1.21e+08	4.55e+04	1.09e+06

Гистограммы непрерывных признаков представлены на рисунках 2–4.

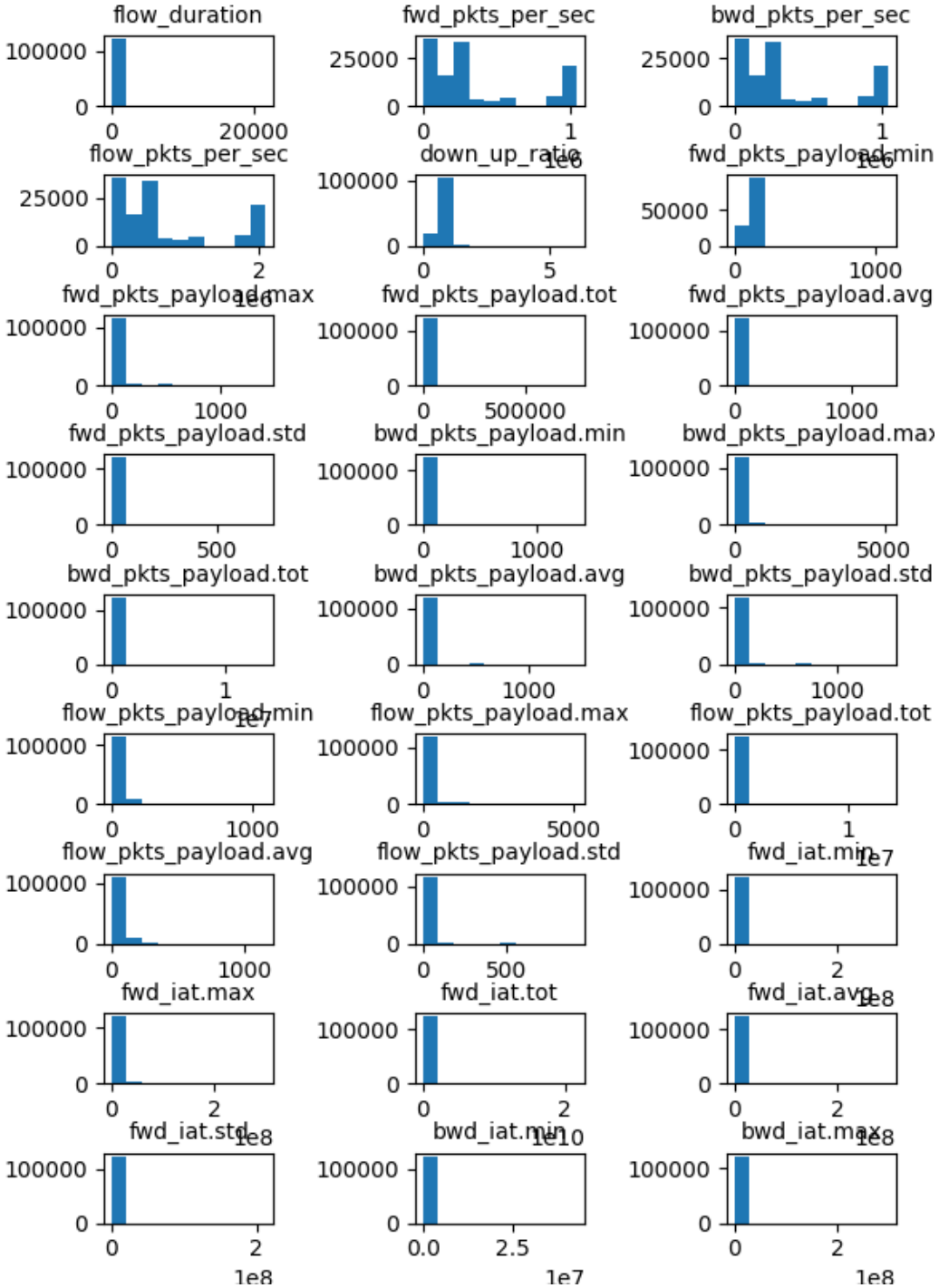

Рисунок 2 – Гистограммы непрерывных признаков (1 часть)

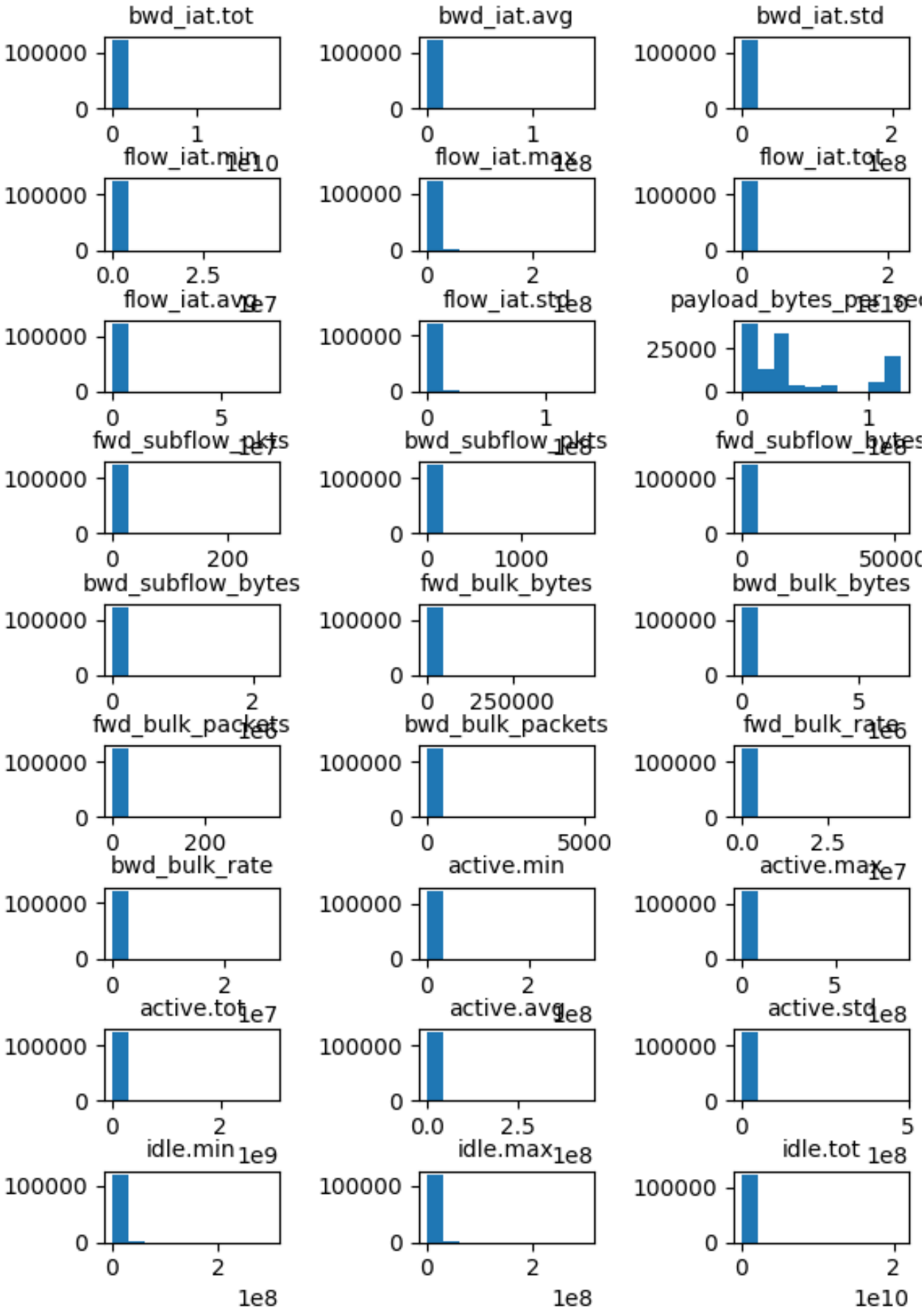

Рисунок 3 – Гистограммы непрерывных признаков (2 часть)

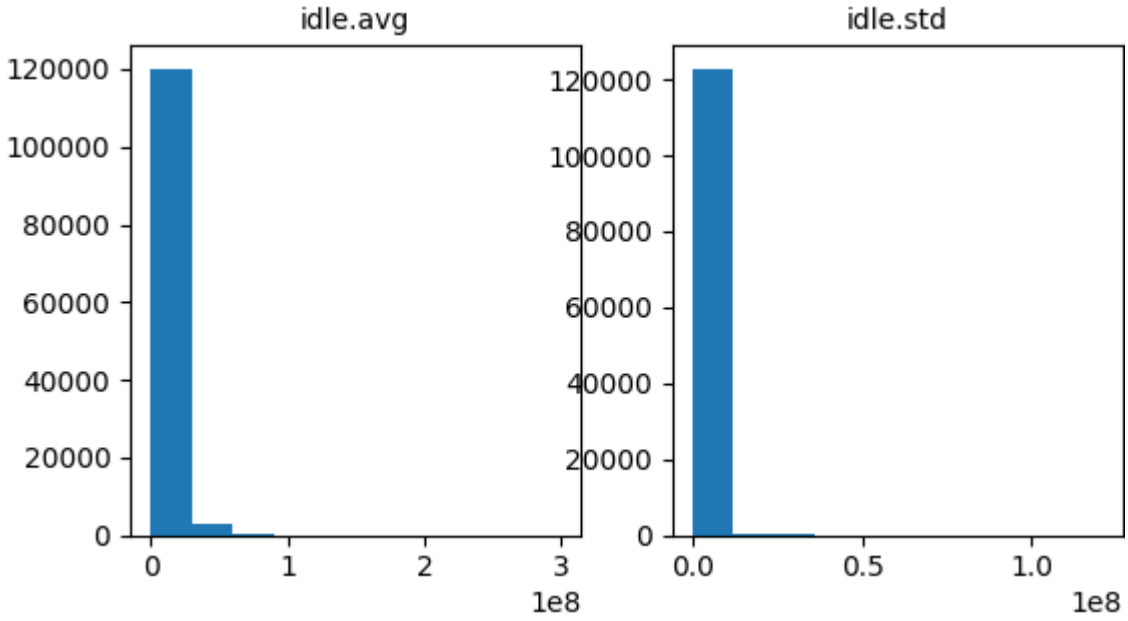

Рисунок 4 – Гистограммы непрерывных признаков (3 часть)

Матрица корреляции для целочисленных и непрерывных признаком представлена на рисунке 5. Красным цветом отображена положительная корреляция, синим – отрицательная, белый обозначает отсутствие корреляции.

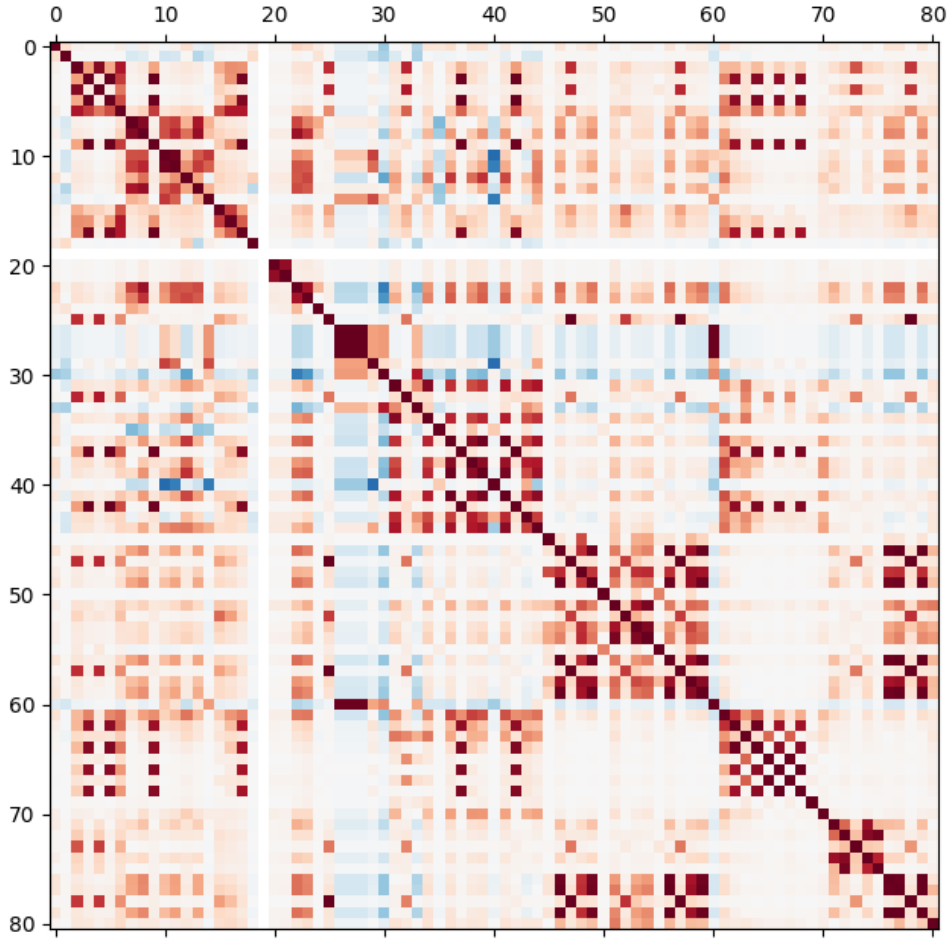

Рисунок 5 – Корреляционная матрица

Проанализируем информацию о признаках. Распределения всех признаков далеки от нормального, диапазоны значений признаков сильно различаются. Для большинства признаков в большей части наблюдений значения близки к 0, однако в остальных наблюдениях значения сильно отличаются от 0.

Признаки коррелируют слабо, однако корреляции все же присутствуют в достаточном количестве.

Заметим, что признак *bwd_URG_flag_count* всегда принимает значение 0 и является по сути бесполезным. Удалим его.

Закодируем категориальные признаки числами. Так, для *proto udp* превратится в 0, *tcp* в 1 и так далее для всех значений всех категориальных признаков.

Выбор методов классификации

Поскольку признаки имеются различного типа (категориальные, целочисленные, непрерывные) и не имеют закон распределения сильно отличается от нормального, байесовский классификатор для таких данных не будет оптимальным решением.

Хорошим решением может стать классифицирующее дерево, поскольку оно может обрабатывать признаки различного типа независимо, не делает никаких предположений об их законе распределения, на него не влияет корреляция признаков.

Еще одним методом, который может быть решением задачи классификации данного набора данных является метод опорных векторов. Он также не делает предположений о нормальном законе распределения, однако чувствителен к масштабу данных, поэтому перед его применением признаки необходимо привести к диапазону.

Классифицирующее дерево

Классифицируем данные при помощи классифицирующего дерева. В ходе подбора выяснилось, что оптимальными параметрами дерева для данного

датасета являются *max_depth*=10, *criterion*='*entropy*', *min_samples_leaf*=1 и *min_samples_split*=2.

Проведем перекрестную проверку классификатора. Для этого разобьём выборку на 3 части и по очереди будем использовать одну из частей для тестирования, а 2 остальные – для обучения. На рисунках 6–8 представлены результаты классификации.

	precision	recall	f1-score	support
ARP_poisioning	0.98	0.98	0.98	2597
DDOS_Slowloris	0.99	0.98	0.99	186
DOS_SYN_Hping	1.00	1.00	1.00	31599
MQTT_Publish	1.00	1.00	1.00	1358
Metasploit_Brute_Force_SSH	0.94	0.79	0.86	19
NMAP_FIN_SCAN	1.00	0.79	0.88	14
NMAP_OS_DETECTION	1.00	1.00	1.00	630
NMAP_TCP_scan	1.00	0.99	1.00	326
NMAP_UDP_SCAN	0.99	0.99	0.99	850
NMAP_XMAS_TREE_SCAN	1.00	1.00	1.00	657
Thing_Speak	0.98	0.99	0.98	2708
Wipro_bulb	0.95	0.95	0.95	95
accuracy			1.00	41039
macro avg	0.99	0.95	0.97	41039
weighted avg	1.00	1.00	1.00	41039

Рисунок 6 – Результат классификации классифицирующим деревом (первое разделение выборки)

	precision	recall	f1-score	support
ARP_poisioning	0.98	0.98	0.98	2568
DDOS_Slowloris	0.99	0.97	0.98	172
DOS_SYN_Hping	1.00	1.00	1.00	31427
MQTT_Publish	1.00	1.00	1.00	1394
Metasploit_Brute_Force_SSH	1.00	0.67	0.80	9
NMAP_FIN_SCAN	1.00	1.00	1.00	6
NMAP_OS_DETECTION	1.00	1.00	1.00	719
NMAP_TCP_scan	1.00	1.00	1.00	344
NMAP_UDP_SCAN	1.00	0.99	0.99	894
NMAP_XMAS_TREE_SCAN	1.00	1.00	1.00	650
Thing_Speak	0.98	0.99	0.98	2767
Wipro_bulb	0.98	0.97	0.97	89
accuracy			1.00	41039
macro avg	0.99	0.96	0.98	41039
weighted avg	1.00	1.00	1.00	41039

Рисунок 7 – Результат классификации классифицирующим деревом (второе разделение выборки)

	precision	recall	f1-score	support
ARP_poisioning	0.98	0.98	0.98	2585
DDOS_Slowloris	0.98	1.00	0.99	176
DOS_SYN_Hping	1.00	1.00	1.00	31633
MQTT_Publish	1.00	1.00	1.00	1394
Metasploit_Brute_Force_SSH	0.88	0.78	0.82	9
NMAP_FIN_SCAN	0.78	0.88	0.82	8
NMAP_OS_DETECTION	1.00	1.00	1.00	651
NMAP_TCP_scan	1.00	0.99	1.00	332
NMAP_UDP_SCAN	1.00	0.98	0.99	846
NMAP_XMAS_TREE_SCAN	1.00	1.00	1.00	703
Thing_Speak	0.98	0.99	0.99	2633
Wipro_bulb	0.94	0.93	0.93	69
accuracy			1.00	41039
macro avg	0.96	0.96	0.96	41039
weighted avg	1.00	1.00	1.00	41039

Рисунок 8 – Результат классификации классифицирующим деревом (третье разделение выборки)

На рисунках 9–11 представлены матрицы ошибок.

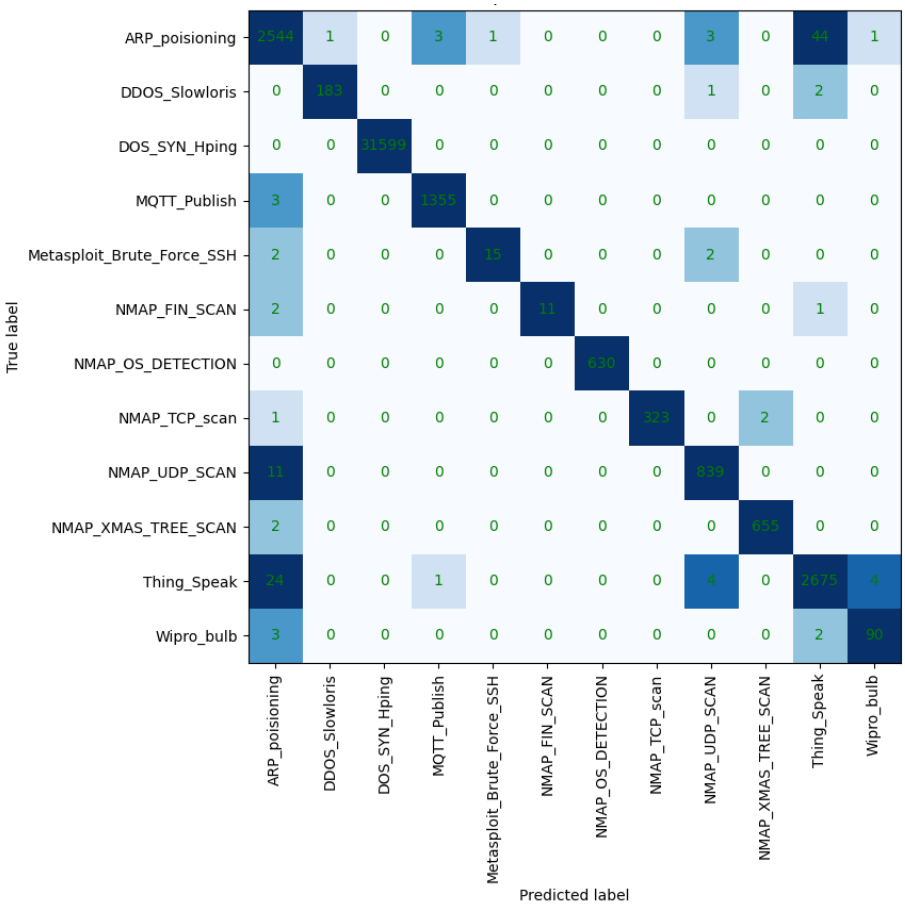

Рисунок 9 – Матрица ошибок классификации классифицирующим деревом (первое разделение выборки)

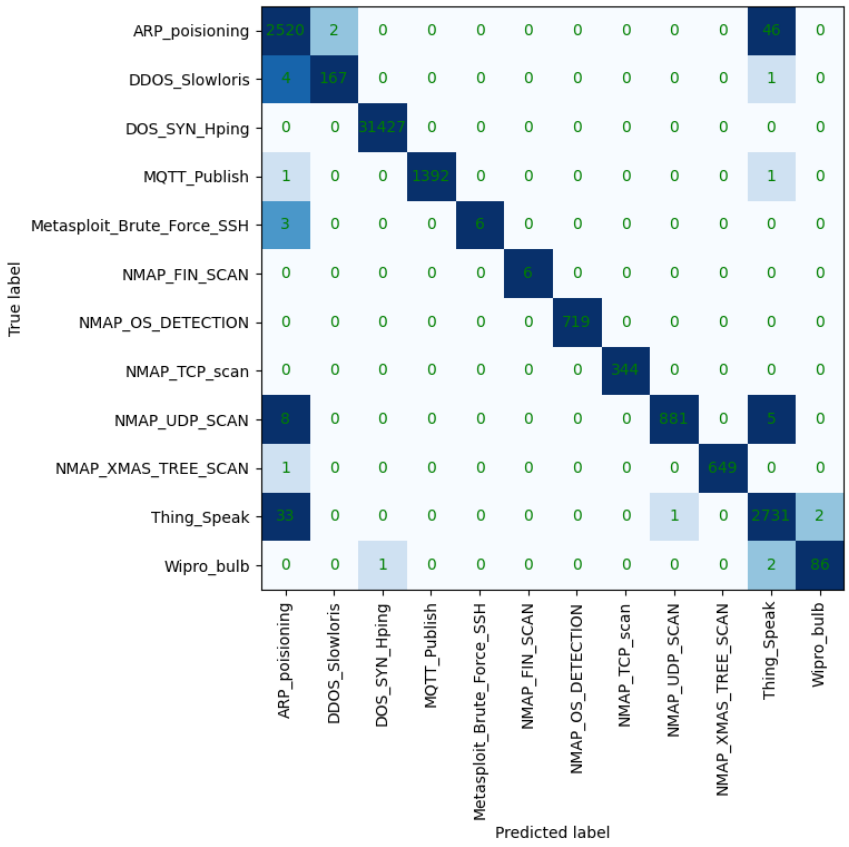

Рисунок 10 – Матрица ошибок классификации классифицирующим деревом (второе разделение выборки)

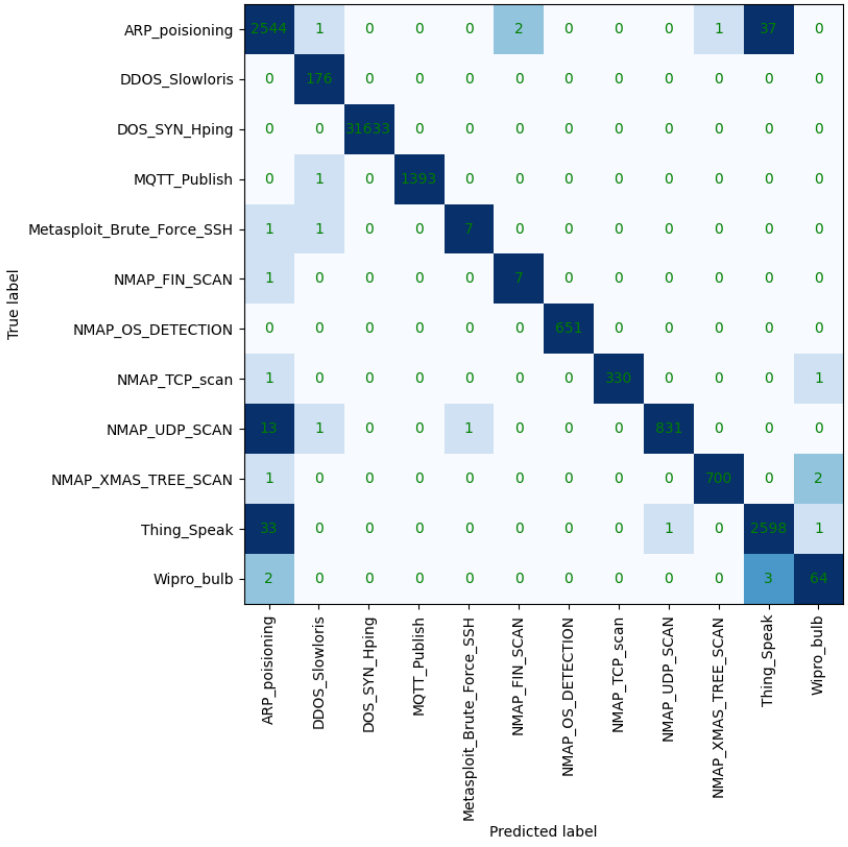

Рисунок 11 – Матрица ошибок классификации классифицирующим деревом (третье разделение выборки)

Проанализируем полученный результат. В целом были получены хорошие результаты: средние значения метрики f1 получились 0.97, 0.98 и 0.96. Однако, если смотреть конкретно, то для классов *Metasploit_Brute_Force_SSH*, *NMAP_FIN_SCAN*, покрытие (recall) часто оказывалось низким. Также из матриц ошибок заметим, что классификатор достаточно часто неправильно предсказывал класс *ARP_poisioning*. Особенно часто он путал *ARP_poisioning* и *Thing_Speak*.

Метод опорных векторов

Классифицируем данные при помощи метода опорных векторов. Поскольку этот метод чувствителен к масштабу признаков, приведем их к диапазону [0;1].

В ходе подбора выяснилось, что оптимальными параметрами классификатора для данного датасета являются *C*=60, *kernel*='*rbf*', *gamma*=2.

Проведем перекрестную проверку классификатора. Для этого разобьём выборку на 3 части и по очереди будем использовать одну из частей для тестирования, а 2 остальные – для обучения. На рисунках 12–14 представлены результаты классификации

	precision	recall	f1-score	support
ARP_poisioning	0.96	0.99	0.98	2533
DDOS_Slowloris	1.00	0.78	0.88	182
DOS_SYN_Hping	1.00	1.00	1.00	31523
MQTT_Publish	1.00	1.00	1.00	1414
Metasploit_Brute_Force_SSH	1.00	1.00	1.00	12
NMAP_FIN_SCAN	1.00	0.92	0.96	12
NMAP_OS_DETECTION	1.00	1.00	1.00	662
NMAP_TCP_scan	1.00	1.00	1.00	327
NMAP_UDP_SCAN	0.95	0.98	0.97	885
NMAP_XMAS_TREE_SCAN	1.00	1.00	1.00	677
Thing_Speak	0.98	0.97	0.98	2731
Wipro_bulb	0.97	0.84	0.90	81
accuracy			1.00	41039
macro avg	0.99	0.96	0.97	41039
weighted avg	1.00	1.00	1.00	41039

Рисунок 12 – Результат классификации методом опорных векторов (первое разделение выборки)

	precision	recall	f1-score	support
ARP_poisioning	0.97	0.98	0.98	2618
DDOS_Slowloris	1.00	0.82	0.90	193
DOS_SYN_Hping	1.00	1.00	1.00	31475
MQTT_Publish	1.00	1.00	1.00	1379
Metasploit_Brute_Force_SSH	1.00	0.75	0.86	12
NMAP_FIN_SCAN	1.00	0.70	0.82	10
NMAP_OS_DETECTION	1.00	1.00	1.00	658
NMAP_TCP_scan	1.00	1.00	1.00	355
NMAP_UDP_SCAN	0.97	0.98	0.97	905
NMAP_XMAS_TREE_SCAN	1.00	1.00	1.00	673
Thing_Speak	0.97	0.98	0.98	2684
Wipro_bulb	0.99	0.90	0.94	77
accuracy			1.00	41039
macro avg	0.99	0.93	0.95	41039
weighted avg	1.00	1.00	1.00	41039

Рисунок 13– Результат классификации методом опорных векторов (второе разделение выборки)

	precision	recall	f1-score	support
ARP_poisioning	0.97	0.98	0.97	2599
DDOS_Slowloris	1.00	0.81	0.90	159
DOS_SYN_Hping	1.00	1.00	1.00	31661
MQTT_Publish	1.00	1.00	1.00	1353
Metasploit_Brute_Force_SSH	1.00	0.54	0.70	13
NMAP_FIN_SCAN	1.00	1.00	1.00	6
NMAP_OS_DETECTION	1.00	1.00	1.00	680
NMAP_TCP_scan	1.00	1.00	1.00	320
NMAP_UDP_SCAN	0.96	0.98	0.97	800
NMAP_XMAS_TREE_SCAN	1.00	1.00	1.00	660
Thing_Speak	0.97	0.98	0.97	2693
Wipro_bulb	1.00	0.83	0.91	95
accuracy			1.00	41039
macro avg	0.99	0.93	0.95	41039
weighted avg	1.00	1.00	1.00	41039

Рисунок 14 – Результат классификации методом опорных векторов (третье разделение выборки)

На рисунках 15–17 представлены матрицы ошибок.

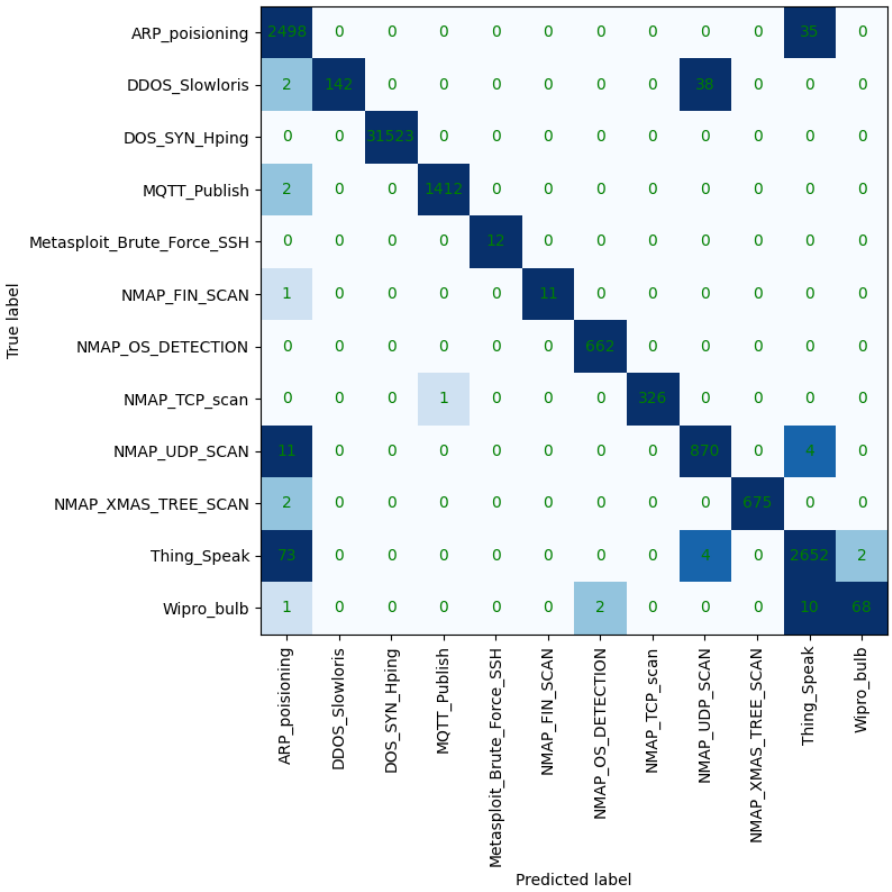

Рисунок 15 – Матрица ошибок классификации методом опорных векторов (первое разделение выборки)

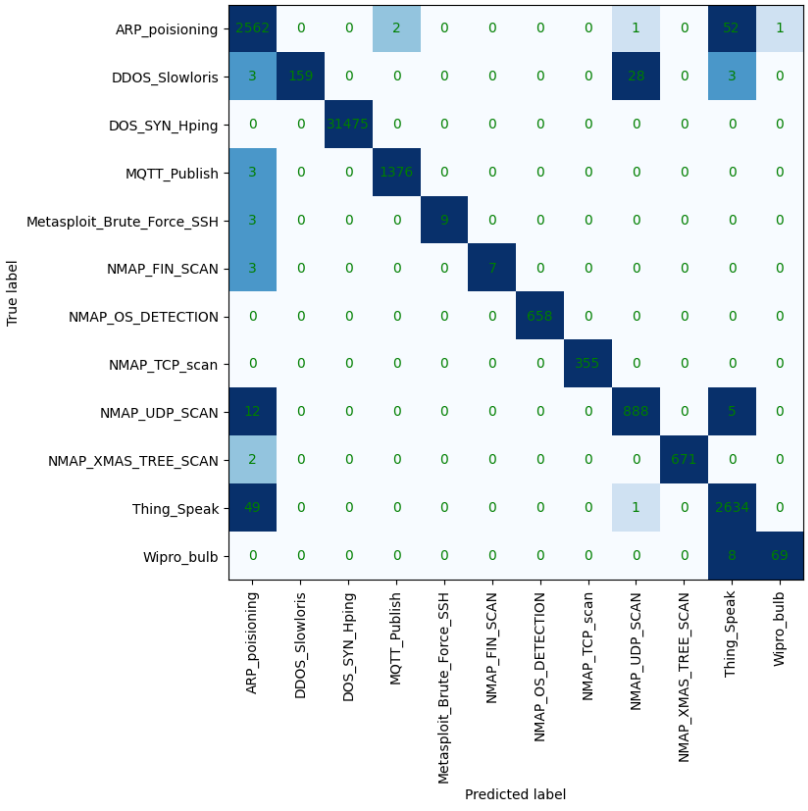

Рисунок 16 – Матрица ошибок классификации методом опорных векторов (второе разделение выборки)

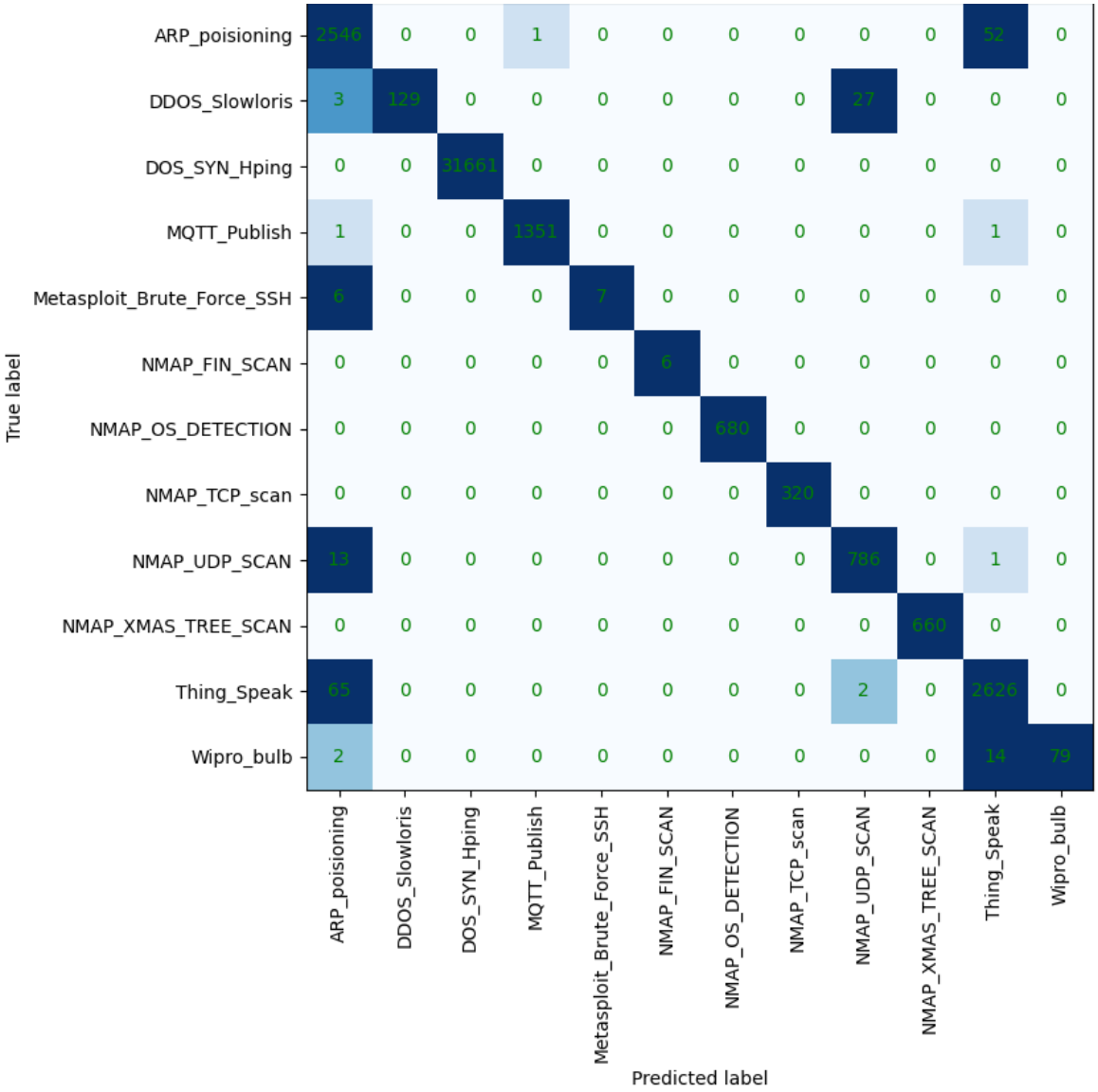

Рисунок 17 – Матрица ошибок классификации методом опорных векторов (третье разделение выборки)

Проанализируем полученный результат. В целом были получены хорошие результаты, однако немного хуже, чем полученные классифицирующим деревом: средние значения метрики f1 получились 0.97, 0.95, 0.95. Проблемы, которые возникали при классификации классифицирующим деревом, оказались актуальными и для этого метода.

Выводы

В ходе выполнения ИДЗ была решена задача классификации типа сетевой атаки по информации о сетевом трафике для датасета «RT-IoT2022».

Для классификации использовались 2 метода: классифицирующее дерево и метод опорных векторов. Методы показали сопоставимый результат, однако классифицирующее дерево оказалось немного лучше по метрике f1. Более того, классифицирующее дерево требует значительно меньше вычислительных ресурсов как для обучения, так и для классификации, поэтому является в данном случае наиболее целесообразным из рассмотренных.

В ходе классификации были выявлены следующие проблемы:

1. низкое покрытие (recall) для классов *Metasploit_Brute_Force_SSH*, *NMAP_FIN_SCAN* (тех классов, наблюдений для которых было меньше всего)*;*
2. ошибочная классификация *ARP_poisioning;*
3. симметричная ошибка в классификации *ARP_poisioning* и *Thing_Speak*.

Перспективными направлениями для дальнейшего улучшения модели является решение проблемы дисбаланса классов, что может решить проблему 1 и более глубокий анализ особенностей классов *ARP_poisioning* и *Thing_Speak*, что может дать решение для проблем 2 и 3.

Код программы, написанной для выполнения ИДЗ представлен в приложениях А, Б, В.

ПРИЛОЖЕНИЕ А

КОД ДЛЯ АНАЛИЗА НАБОРА ДАННЫХ

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv('RT_IOT2022.csv',index_col=0)
print(df)
print(df.describe())
print(df['Attack_type'].value_counts())

print(df.dtypes)
categ_cols=df.select_dtypes(include=['object']).columns.tolist()
num_cols = df.select_dtypes(include=['int64']).columns.tolist()
cont_cols=df.select_dtypes(include=['float64']).columns.tolist()

dfcat=df[categ_cols]
dfnum=df[num_cols]
dfcon=df[cont_cols]

for cat in categ_cols:
    print(cat,set(dfcat[cat].tolist()))

print(dfnum.iloc[:, :10].describe())
print(dfnum.iloc[:, 10:20].describe())
print(dfnum.iloc[:, 20:].describe())

print(dfcon.iloc[:, :10].describe())
print(dfcon.iloc[:, 10:20].describe())
print(dfcon.iloc[:, 20:30].describe())
print(dfcon.iloc[:, 30:40].describe())
print(dfcon.iloc[:, 40:50].describe())
print(dfcon.iloc[:, 50:].describe())

fig, axs = plt.subplots(len(num_cols)//3+1,3)
for i in range(len(num_cols)):
    axs[i//3, i%3].hist(dfnum.to_numpy()[:,i], bins = 10)
    axs[i//3, i%3].set_title(dfnum.columns[i],{'fontsize':10})
plt.show()
fig, axs = plt.subplots(9,3)
for i in range(27):
    axs[i//3, i%3].hist(dfcon.to_numpy()[:,i], bins = 10)
    axs[i//3, i%3].set_title(dfcon.columns[i],{'fontsize':10})
plt.show()
fig, axs = plt.subplots(9,3)
for i in range(27):
    axs[i//3, i%3].hist(dfcon.to_numpy()[:,i+27], bins = 10)
    axs[i//3, i%3].set_title(dfcon.columns[i+27],{'fontsize':10})
plt.show()
fig, axs = plt.subplots(2,2)
for i in range(len(cont_cols)-54):
    axs[i//3, i%3].hist(dfcon.to_numpy()[:,i+54], bins = 10)
    axs[i//3, i%3].set_title(dfcon.columns[i+54],{'fontsize':10})
plt.show()
dfn=df[num_cols+cont_cols]
corrmat=dfn.corr()
plt.matshow(corrmat, cmap='RdBu_r', vmin=-1, vmax=1)
plt.show()
```

ПРИЛОЖЕНИЕ Б

КОД КЛАССИФИКАЦИИ ПРИ ПОМОЩИ КЛАССИФИЦИРУЮЩЕГО ДЕРЕВА

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn import metrics
from sklearn import tree
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn import metrics
from sklearn import preprocessing

df = pd.read_csv("RT_IOT2022.csv",index_col=0)

data = df.drop(columns=['Attack_type','bwd_URG_flag_count'])
labels=df['Attack_type']

categ_cols=data.select_dtypes(include=['object']).columns.tolist()
for cat in categ_cols:
    le = preprocessing.LabelEncoder()
    data[cat]= le.fit_transform(data[cat])

le = preprocessing.LabelEncoder()
labels= le.fit_transform(labels)

data_split=[None,None,None]
labels_split=[None,None,None]
data_split[0], data_split[1], labels_split[0], labels_split[1] =
model_selection.train_test_split(data, labels, test_size=1/3)
data_split[0], data_split[2], labels_split[0], labels_split[2] =
model_selection.train_test_split(data_split[0], labels_split[0],
test_size=0.5)

for i in range(3):
    data_test=data_split[i]
    labels_test=labels_split[i]
    data_train = pd.concat(data_split[:i]+data_split[i+1:], axis=0)
    labels_train = np.concatenate(labels_split[:i]+labels_split[i+1:],
axis=0)

    clf = tree.DecisionTreeClassifier(max_depth=10,criterion='entropy')
    labels_pred = clf.fit(data_train,labels_train).predict(data_test)

    print(metrics.classification_report(labels_test,
labels_pred,target_names=le.classes_))

    cm = metrics.confusion_matrix(labels_test,labels_pred)
    cmdplot = metrics.ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=le.classes_)
    cmdplot.plot(cmap='Blues', text_kw={'color': 'Green'},xticks_rotation=90)
    cmdplot.im_.set_clim(vmin=0, vmax=5)
    plt.title('Матрица ошибок')
    plt.show()
```

ПРИЛОЖЕНИЕ В

КОД ДЛЯ КЛАССИФИКАЦИИ ПРИ ПОМОЩИ МЕТОДА ОПОРНЫХ ВЕКТОРОВ

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn import metrics
from sklearn import svm
from sklearn import preprocessing
df = pd.read_csv("RT_IOT2022.csv",index_col=0)

data = df.drop(columns=['Attack_type','bwd_URG_flag_count'])
labels=df['Attack_type']

categ_cols=data.select_dtypes(include=['object']).columns.tolist()
for cat in categ_cols:
    le = preprocessing.LabelEncoder()
    data[cat]= le.fit_transform(data[cat])

le = preprocessing.LabelEncoder()
labels= le.fit_transform(labels)

scaler = preprocessing.MinMaxScaler()
data[data.columns] = scaler.fit_transform(data[data.columns])

data_split=[None,None,None]
labels_split=[None,None,None]
data_split[0], data_split[1], labels_split[0], labels_split[1] =
model_selection.train_test_split(data, labels, test_size=1/3)
data_split[0], data_split[2], labels_split[0], labels_split[2] =
model_selection.train_test_split(data_split[0], labels_split[0],
test_size=0.5)

for i in range(3):
    data_test=data_split[i]
    labels_test=labels_split[i]
    data_train = pd.concat(data_split[:i]+data_split[i+1:], axis=0)
    labels_train = np.concatenate(labels_split[:i]+labels_split[i+1:],
axis=0)

    clf = svm.SVC(kernel='rbf',C=60,gamma=2)
    labels_pred = clf.fit(data_train,labels_train).predict(data_test)

    print(metrics.classification_report(labels_test,
labels_pred,target_names=le.classes_))

    cm = metrics.confusion_matrix(labels_test,labels_pred)
    cmdplot = metrics.ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=le.classes_)
    cmdplot.plot(cmap='Blues', text_kw={'color': 'Green'},xticks_rotation=90)
    cmdplot.im_.set_clim(vmin=0, vmax=5)
    plt.title('Матрица ошибок')
    plt.show()
```