Детекция кибератак методами машинного обучения

Корнеев Герман
Печенкина Наталья

Студенты 4 курса

15.10.2025

Проблема и цель проекта

Проблема: экспоненциальный рост объемов сетевого трафика и киберугроз делает ручной анализ невозможным и требует автоматизированных систем

Цель: разработать модель для бинарной классификации сетевого трафика («норма» vs «атака») с использованием технологий Big Data

Датасет: CIC-IDS-2017 (Canadian Institute for Cybersecurity Intrusion Detection System 2017) (>2.3 млн записей)

Проблема и цель проекта



Структура CIC-IDS-2017

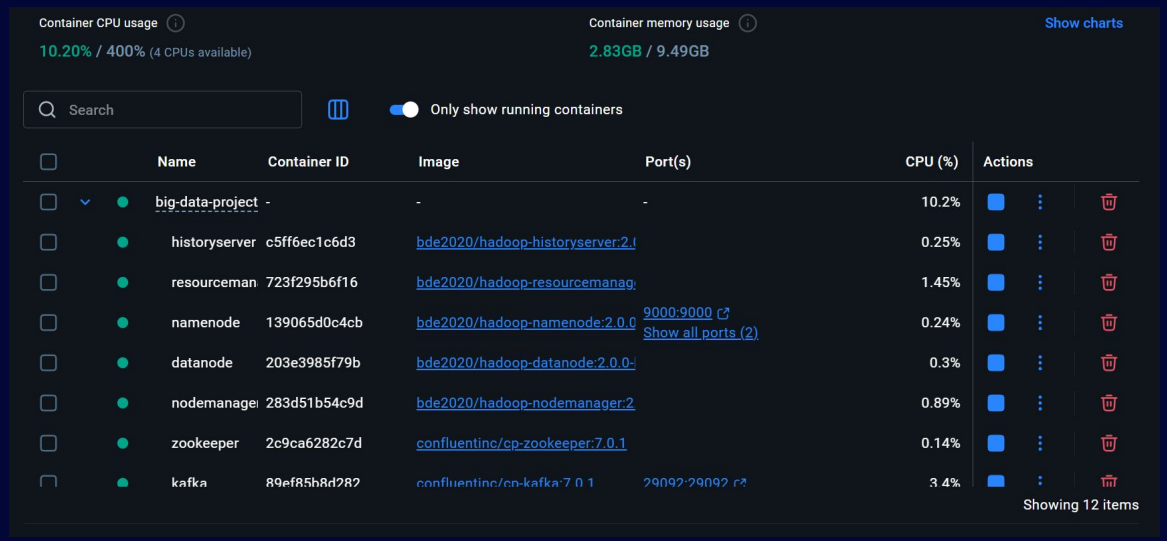


CIC-IDS-2017

```
|-- Protocol: integer (nullable = true)
|-- Flow Duration: long (nullable = true)
|-- Total Fwd Packets: integer (nullable = true)
|-- Total Backward Packets: integer (nullable = true)
|-- Fwd Packets Length Total: long (nullable = true)
|-- Bwd Packets Length Total: long (nullable = true)
|-- Fwd Packet Length Max: integer (nullable = true)
|-- Fwd Packet Length Min: integer (nullable = true)
|-- Fwd Packet Length Mean: float (nullable = true)
|-- Fwd Packet Length Std: float (nullable = true)
|-- Bwd Packet Length Max: integer (nullable = true)
|-- Bwd Packet Length Min: integer (nullable = true)
|-- Bwd Packet Length Mean: float (nullable = true)
|-- Bwd Packet Length Std: float (nullable = true)
|-- Flow Bytes/s: double (nullable = true)
|-- Flow Packets/s: double (nullable = true)
|-- Flow IAT Mean: double (nullable = true)
|-- Flow IAT Std: double (nullable = true)
|-- Flow IAT Max: long (nullable = true)
|-- Flow IAT Min: long (nullable = true)
|-- Fwd IAT Total: long (nullable = true)
|-- Fwd IAT Mean: double (nullable = true)
|-- Fwd IAT Std: double (nullable = true)
|-- Fwd IAT Max: long (nullable = true)
|-- Fwd IAT Min: long (nullable = true)
|-- Bwd IAT Total: long (nullable = true)
|-- Bwd IAT Mean: double (nullable = true)
|-- Bwd IAT Std: double (nullable = true)
|-- Bwd IAT Max: long (nullable = true)
|-- Bwd IAT Min: long (nullable = true)
|-- Fwd PSH Flags: integer (nullable = true)
|-- Bwd PSH Flags: integer (nullable = true)
|-- Fwd URG Flags: integer (nullable = true)
|-- Bwd URG Flags: integer (nullable = true)
|-- Fwd Header Length: long (nullable = true)
|-- Bwd Header Length: long (nullable = true)
|-- Fwd Packets/s: double (nullable = true)
|-- Bwd Packets/s: double (nullable = true)
|-- Packet Length Min: integer (nullable = true)
|-- Packet Length Max: integer (nullable = true)
|-- Packet Length Mean: float (nullable = true)
|-- Packet Length Std: float (nullable = true)
|-- Packet Length Variance: float (nullable = true)
```

```
|-- Packet Length Variance: float (nullable = true)
|-- FIN Flag Count: integer (nullable = true)
|-- SYN Flag Count: integer (nullable = true)
|-- RST Flag Count: integer (nullable = true)
|-- PSH Flag Count: integer (nullable = true)
|-- ACK Flag Count: integer (nullable = true)
|-- URG Flag Count: integer (nullable = true)
|-- CWE Flag Count: integer (nullable = true)
|-- ECE Flag Count: integer (nullable = true)
|-- Down/Up Ratio: integer (nullable = true)
|-- Avg Packet Size: float (nullable = true)
|-- Avg Fwd Segment Size: float (nullable = true)
|-- Avg Bwd Segment Size: float (nullable = true)
|-- Fwd Avg Bytes/Bulk: integer (nullable = true)
|-- Fwd Avg Packets/Bulk: integer (nullable = true)
|-- Fwd Avg Bulk Rate: integer (nullable = true)
|-- Bwd Avg Bytes/Bulk: integer (nullable = true)
|-- Bwd Avg Packets/Bulk: integer (nullable = true)
|-- Bwd Avg Bulk Rate: integer (nullable = true)
|-- Subflow Fwd Packets: integer (nullable = true)
|-- Subflow Fwd Bytes: long (nullable = true)
|-- Subflow Bwd Packets: integer (nullable = true)
|-- Subflow Bwd Bytes: long (nullable = true)
|-- Init Fwd Win Bytes: integer (nullable = true)
|-- Init Bwd Win Bytes: integer (nullable = true)
|-- Fwd Act Data Packets: integer (nullable = true)
|-- Fwd Seg Size Min: integer (nullable = true)
|-- Active Mean: double (nullable = true)
|-- Active Std: double (nullable = true)
|-- Active Max: long (nullable = true)
|-- Active Min: long (nullable = true)
|-- Idle Mean: double (nullable = true)
|-- Idle Std: double (nullable = true)
|-- Idle Max: long (nullable = true)
|-- Idle Min: long (nullable = true)
|-- Label: string (nullable = true)
```

Архитектура решения




```
PS D:\Docker\Big-Data-Project> docker-compose up -d
[+] Running 12/12
 ✔Network big-data-project_hadoop-spark-network  Created
 ✔Container resourcemanager                     Started
 ✔Container nodemanager                         Started
 ✔Container namenode                            Started
 ✔Container datanode                            Started
 ✔Container historyserver                       Started
 ✔Container zookeeper                           Started
 ✔Container spark-master                        Started
 ✔Container kafka                               Started
 ✔Container spark-worker                        Started
 ✔Container jupyter                             Started
 ✔Container traffic-producer                    Started
PS D:\Docker\Big-Data-Project>
```

HDFS

распределенное хранилище данных

Jupyter

интерактивная среда разработки

YARN

диспетчер ресурсов

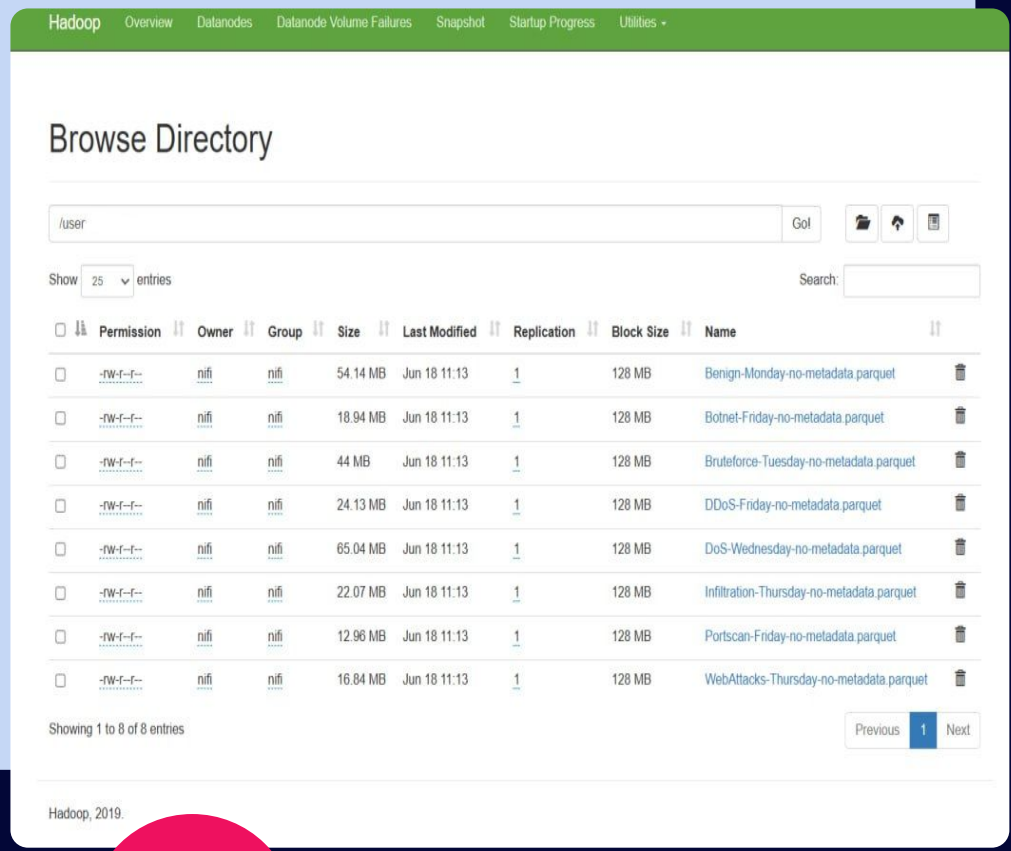
Kafka

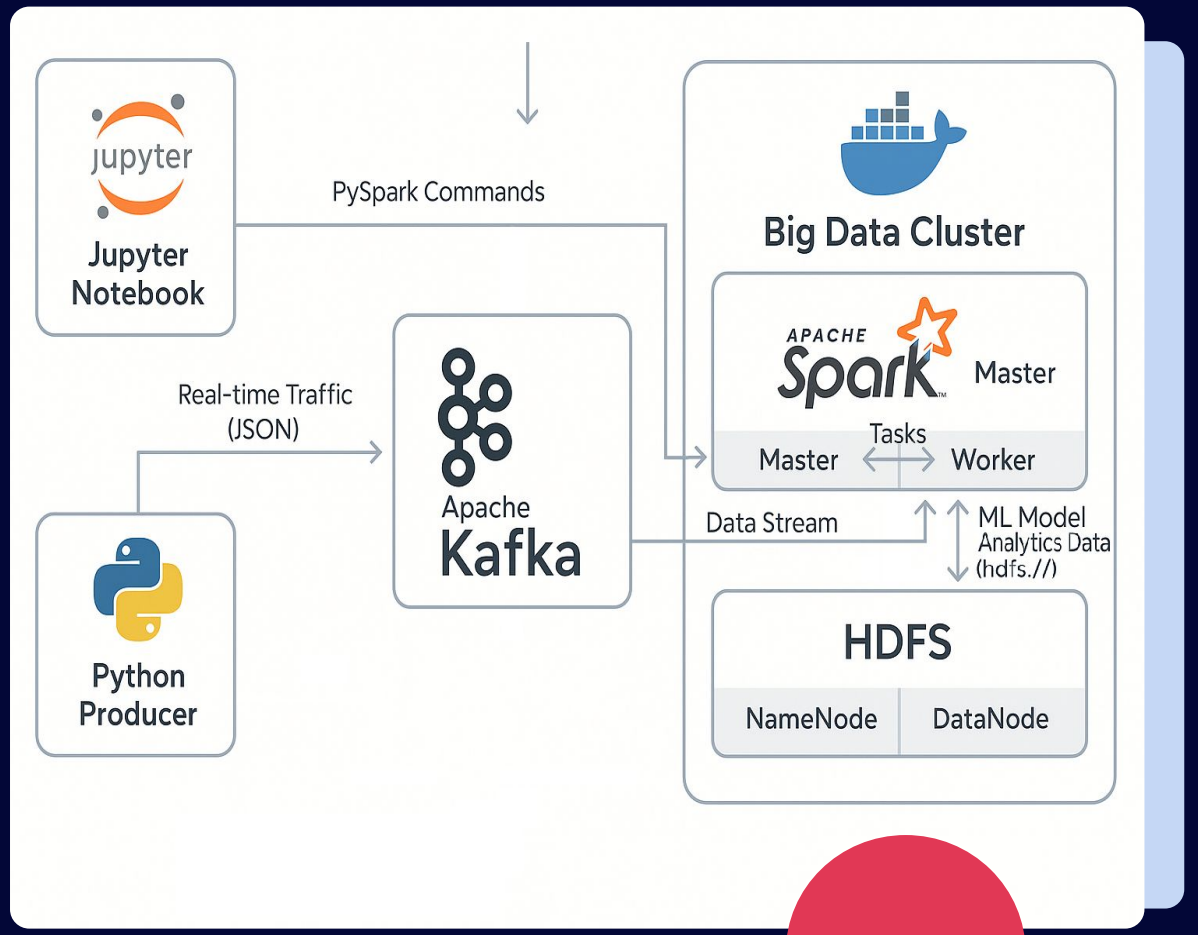
диспетчер потоковых данных

Apache Spark

ядро системы

Архитектура решения





Подготовка и очистка данных

Проблема: неконсистентность схем в Parquet-файлах

Решение: разработка и применение единой, эталонной схемы данных

Проблема: некорректные символы в названиях колонок из-за ошибок кодировки

Решение: замена символов и переименование колонок для приведения к читаемому виду

Проблема: наличие NULL, NaN и аномальных значений и некорректных меток

Решение: фильтрация и очистка данных

```
_path = "hdfs://namenode:9000/data/cicids2017/"

es = [
enign-Monday-no-metadata.parquet",
"Botnet-Friday-no-metadata.parquet",
"Bruteforce-Tuesday-no-metadata.parquet",
"DDoS-Friday-no-metadata.parquet",
"DoS-Wednesday-no-metadata.parquet",
"Infiltration-Thursday-no-metadata.parquet",
"Portscan-Friday-no-metadata.parquet",
"WebAttacks-Thursday-no-metadata.parquet"
]

parquet_files = [hdfs_base_path + name for name in file_names]
print(f"Будет обработано {len(parquet_files)} файлов из ручного списка.")


# Читаем каждый файл, приводим к эталонной схеме и добавляем в список
all_dfs = []
for file_path in parquet_files:
    try:
        print(f"Обработка файла: {file_path}...")
        temp_df = spark.read.parquet(file_path)

        select_expr = []
        for field in manual_schema.fields:
            col_name = f"`{field.name}`"
            target_type = field.dataType
            select_expr.append(F.col(col_name).cast(target_type).alias(field.name))

        conformed_df = temp_df.select(select_expr)
        all_dfs.append(conformed_df)
        print(f"Файл {file_path} успешно обработан.")
    except Exception as e:
        print(f"Ошибка при обработке файла {file_path}: {e}")


# Объединяем все DataFrame'ы в один
if not all_dfs:
    raise ValueError("Не удалось обработать ни одного файла.")
else:
    print("\nОбъединение всех данных в один DataFrame...")
    df = reduce(DataFrame.unionByName, all_dfs)
    print("Все файлы успешно прочитаны и объединены!")

    df.printSchema()
```

Основные статистические показатели для ключевых числовых признаков:

summary	Flow Duration	Total Fwd Packets	Total Backward Packets	Fwd Packet Length Max	Bwd Packet Length Max	Flow IAT Mean
count	2,313,810.00	2,313,810.00	2,313,810.00	2,313,810.00	2,313,810.00	2,313,810.00
mean	18,075,748.00	11.08	12.55	250.63	1,063.22	1,575,398.75
stddev	36,408,232.00	829.19	1,103.18	785.51	2,105.20	4,866,184.00
min	-13.00	1.00	0.00	0.00	0.00	-13.00
max	120,000,000.00	219,759.00	291,922.00	24,820.00	19,530.00	120,000,000.00

Исследование данных

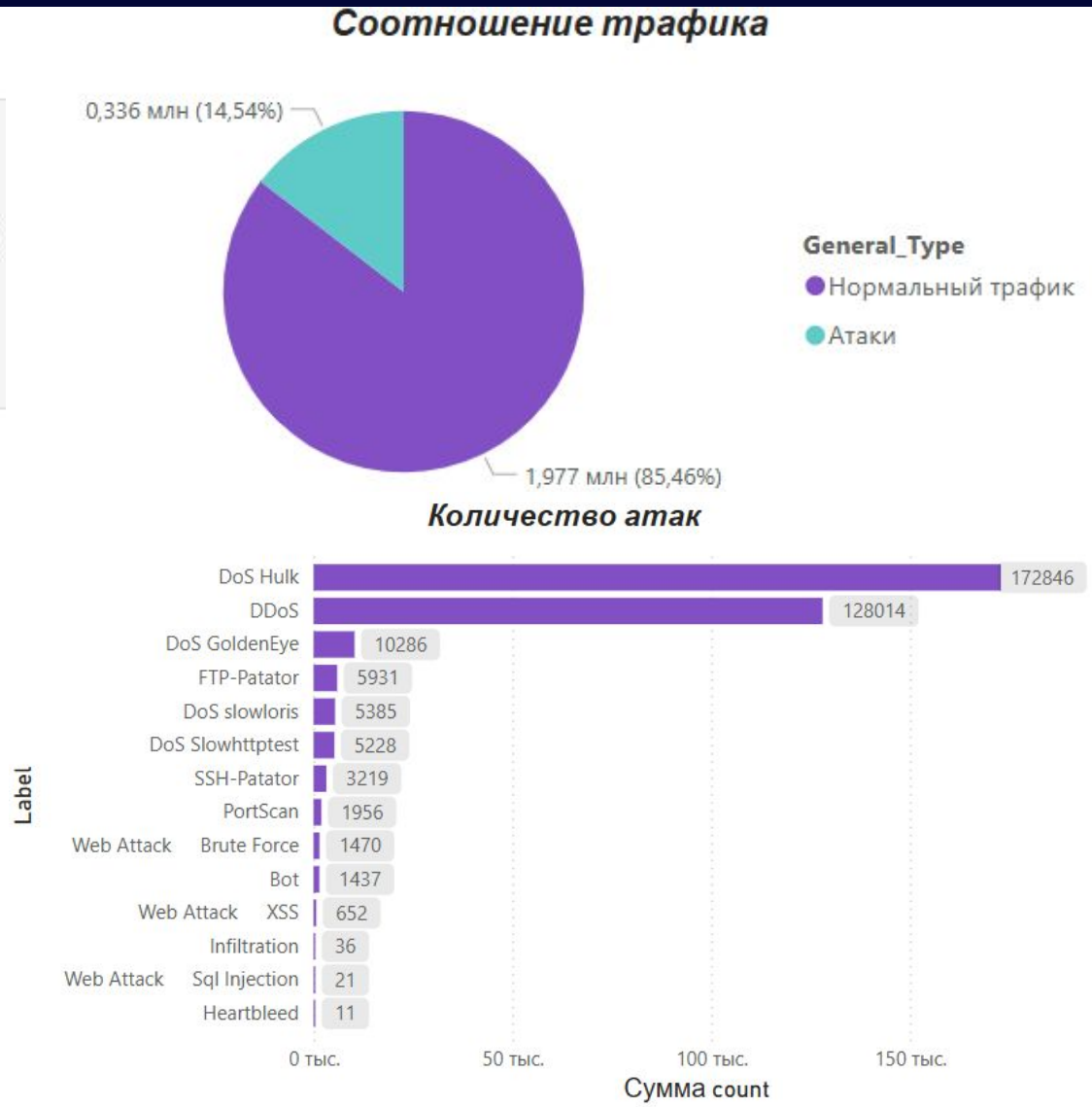
```
Подсчет общего количества записей...
Общее количество записей в датасете: 2313810
```

```
# Анализ меток (Labels)
print("Анализ распределения классов (меток 'Label'):")

label_counts = df_cleaned.groupBy("Label").count().orderBy("count", ascending=False)
label_counts.show(20, truncate=False)

# Этот DataFrame для Power BI
label_counts_for_bi = label_counts
```

```
Анализ распределения классов (меток 'Label'):
+--------------------------+-------+
|Label                     |count  |
+--------------------------+-------+
|Benign                    |1977318|
|DoS Hulk                  |172846 |
|DDoS                      |128014 |
|DoS GoldenEye             |10286  |
|FTP-Patator               |5931   |
|DoS slowloris             |5385   |
|DoS Slowhttptest          |5228   |
|SSH-Patator               |3219   |
|PortScan                  |1956   |
|Web Attack - Brute Force  |1470   |
|Bot                       |1437   |
|Web Attack - XSS          |652    |
|Infiltration              |36     |
|Web Attack - Sql Injection|21     |
|Heartbleed                |11     |
+--------------------------+-------+
```



Анализ поведенческих паттернов

Гипотеза: *вредоносный сетевой трафик имеет статистически значимые отличия от нормального трафика по своим временным и объемным характеристикам; при этом можно предположить, что разные типы атак будут демонстрировать разные, уникальные поведенческие паттерны.*

```
--- Проверка гипотезы: Вредоносный трафик статистически отличается от нормального ---

Начало сравнительного анализа ключевых признаков...
Сравнительная таблица средних значений (Норма vs Атака):
+------------+--------------------+---------------------+--------------------------+-----------------+------------------+
|traffic_type|avg_Flow_Duration   |avg_Total_Fwd_Packets|avg_Total_Backward_Packets|avg_Flow_Bytes_s |avg_Flow_IAT_Mean |
+------------+--------------------+---------------------+--------------------------+-----------------+------------------+
|Норма       |1.2884191235363761E7|11.96069979639087    |13.846112259130802        |570250.7395219771|1029699.8064273001|
|Атака       |4.85827415104579E7  |5.931683368401032    |4.95664978662197          |65783.96028591892|4782073.582414234 |
+------------+--------------------+---------------------+--------------------------+-----------------+------------------+

Гипотеза о различии характеристик трафика подтверждена численно.

Начало детальной проверки гипотезы...
Сравнительная таблица для проверки гипотезы:
+-----------------------+--------------------+------------------+------------------+------------------+
|Label                  |avg_Flow_Duration   |avg_Fwd_Packets   |avg_Bwd_Packets   |avg_Flow_Bytes_s  |
+-----------------------+--------------------+------------------+------------------+------------------+
|Benign                 |1.2884191235363761E7|11.96069979639087 |13.846112259130802|570250.7395219771 |
|DDoS                   |1.6957157952700485E7|4.472682675332385 |3.2560813661005827|60509.88892040267 |
|DoS Hulk               |7.6310925088170 98E7|6.363300278860951 |5.620164770952177 |37924.139208246794|
|Infiltration           |7.84077205E7        |830.2222222222222 |829.6111111111111 |20188.219030217413|
|PortScan               |6725413.786298568   |2.425357873210634 |1.4028629856850716|137852.7159959749 |
|Web Attack - Brute Force|6669941.34217687   |12.657823129251701|6.183673469387755 |183.79325089718367|
+-----------------------+--------------------+------------------+------------------+------------------+

Проверка гипотезы завершена.
```

Логистическая регрессия vs Случайный лес

Логистическая регрессия

- Линейная модель
- Быстрая в обучении
- Хорошо интерпретируема

```python
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import StandardScaler
from pyspark.ml import Pipeline

# Создаем StandardScaler для масштабирования признаков
scaler = StandardScaler(inputCol="features", outputCol="scaledFeatures", withStd=True, withMean=True)

# Модель будет работать с отмасштабированными признаками 'scaledFeatures'
lr = LogisticRegression(labelCol="label_index", featuresCol="scaledFeatures")

# Собираем конвейер: индексатор -> векторизатор -> масштабировщик -> модель
lr_pipeline = Pipeline(stages=[label_indexer, vector_assembler, scaler, lr])

print("Начинаем обучение улучшенной модели 'Логистическая регрессия' (с масштабированием)...")
start_time_lr = time.time()
lr_pipeline_model = lr_pipeline.fit(training_data)
end_time_lr = time.time()
lr_training_time = end_time_lr - start_time_lr
print(f"Модель успешно обучена! Время обучения: {lr_training_time:.2f} секунд.")
```

```
Начинаем обучение улучшенной модели 'Логистическая регрессия' (с масштабированием)...
Модель успешно обучена! Время обучения: 155.18 секунд.
```

Случайный лес

- Ансамбль решающих деревьев
- Обеспечивает высокую точность
- Устойчив к переобучению

```python
from pyspark.ml import Pipeline
from pyspark.ml.classification import RandomForestClassifier

# СОЗДАНИЕ МОДЕЛИ
# Мы используем Случайный лес. labelCol и featuresCol - стандартные имена,
# которые мы задали в наших трансформерах
rf = RandomForestClassifier(labelCol="label_index", featuresCol="features", numTrees=100, seed=42)

# СБОРКА КОНВЕЙЕРА (PIPELINE)
# Мы объединяем label_indexer, vector_assembler и модель (rf)
# в единую последовательность шагов.
# Spark будет автоматически выполнять их один за другим
pipeline = Pipeline(stages=[label_indexer, vector_assembler, rf])

# ОБУЧЕНИЕ МОДЕЛИ
print("Начинаем обучение модели 'Случайный лес'...")

start_time = time.time()

# Команда .fit() запускает процесс обучения Pipeline на обучающих данных.
# Spark последовательно применит StringIndexer, VectorAssembler, а затем обучит RandomForest
pipeline_model = pipeline.fit(training_data)

end_time = time.time()
training_time = end_time - start_time

print(f"Модель успешно обучена! Время обучения: {training_time:.2f} секунд.")
```

```
Начинаем обучение модели 'Случайный лес'...
Модель успешно обучена! Время обучения: 230.05 секунд.
```

Визуализация с использованием Power BI

Записей в DataSet
2 313 810

Норма **1 977 318** Атак **336 492**

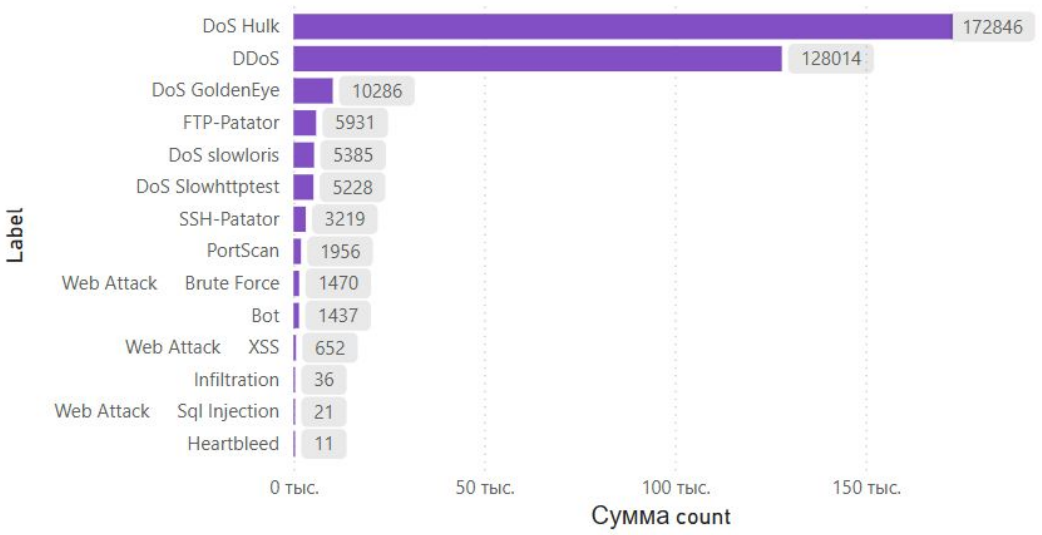
Случайный лес

Accuracy	Precision	Recall
98.31%	**99.50%**	**88.85%**

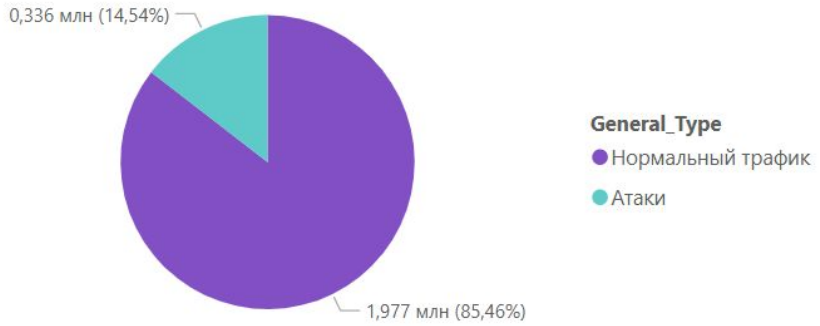
Логистическая регрессия

Accuracy	Precision	Recall
97.07%	**97.61%**	**81.87%**

Количество атак



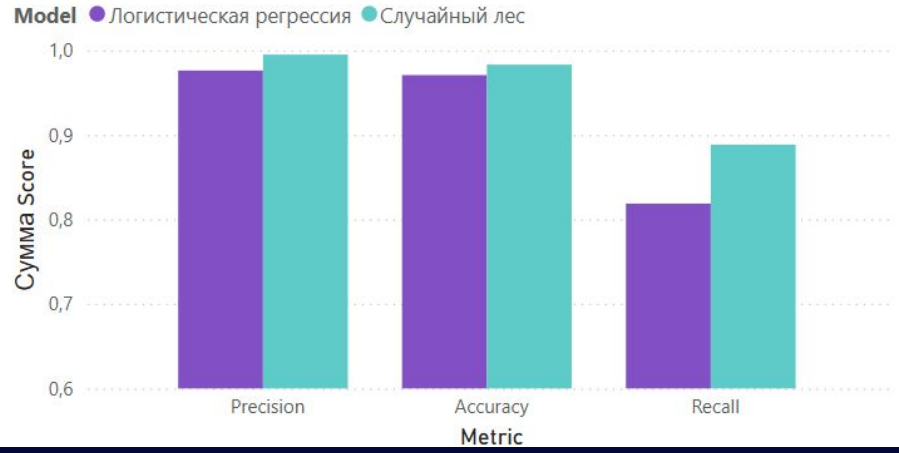
Соотношение трафика



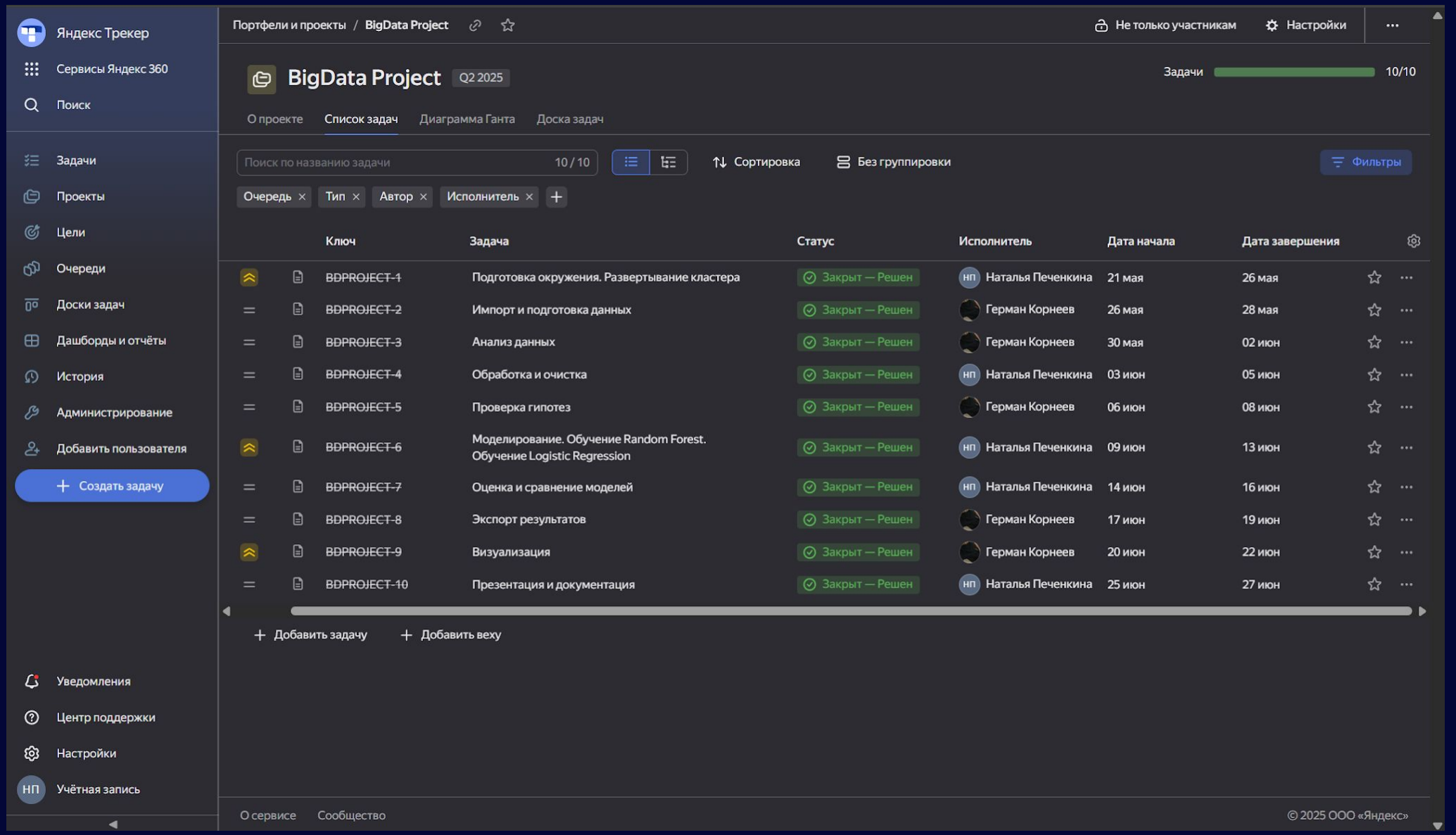
Real_Label	Атака	Норма	Всего
Атака	60134	7132	**67266**
Норма	618	395747	**396365**
Всего	**60752**	**402879**	**463631**

Label	Сумма count
DoS Hulk	172846
DDoS	128014
DoS GoldenEye	10286
FTP-Patator	5931
DoS slowloris	5385
Всего	**322462**

Сравнение производительности моделей



Организация командной работы



Спасибо за внимание!

Корнеев Герман
Печенкина Наталья

Студенты 4 курса

Материалы проекта