Виявлення шахрайських транзакцій в потокових даних

Виконав:
студент групи ІП-43мп
Гриценко Андрій

Вступ

Мета роботи:

- Розробка системи для виявлення шахрайських транзакцій у режимі near real-time з використанням Apache Spark Streaming.

Основні завдання:

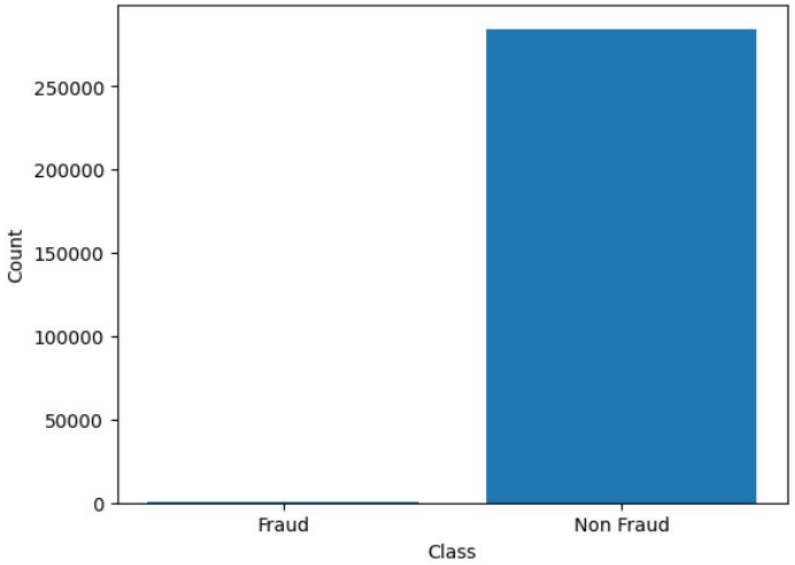
- Аналіз ефективності виявлення шахрайства за допомогою різних алгоритмів.
- Дослідження швидкодії системи за різних сценаріїв навантаження.

Джерело даних:

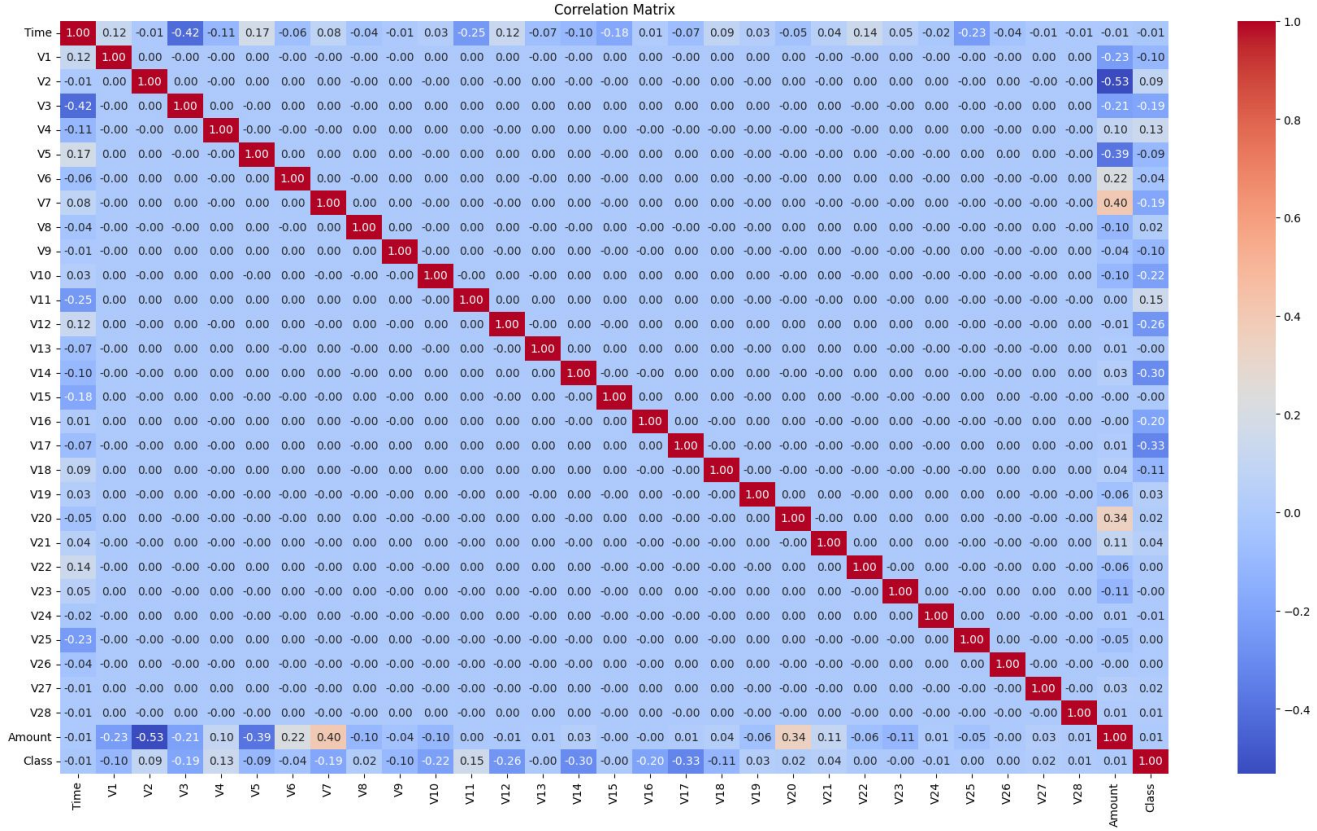
- Dataset: "Credit Card Fraud Detection"
- Платформа: Kaggle
- URL: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

Огляд набору даних

1. Time (час) - кількість секунд, які пройшли з моменту першої транзакції у наборі даних.
2. V1 до V28 - анонімізовані числові ознаки, отримані за допомогою аналізу головних компонент (PCA).
3. Amount (сума) - відображає грошову суму транзакції, без прив’язки до валюти
4. Class (Клас) - цільова мітка, (1 - шахрайська) та 0 - звичайна транзакція



Матриця кореляції



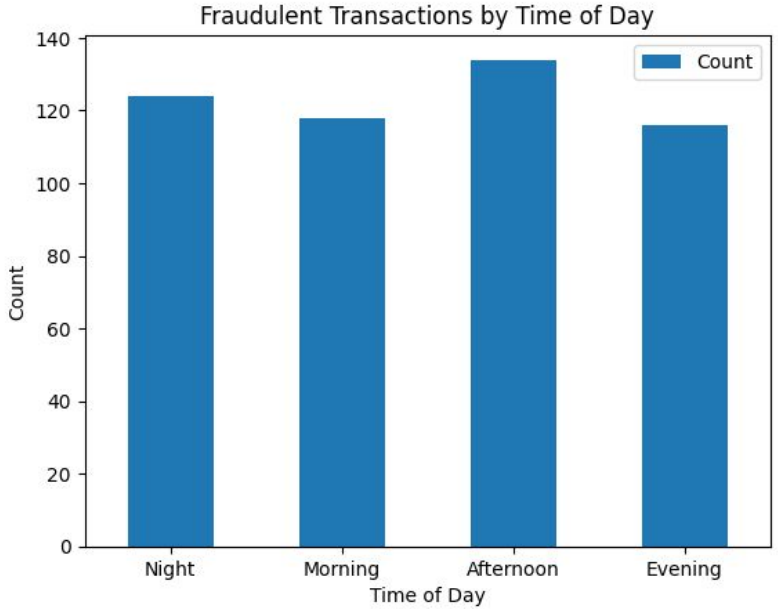
Огляд набору даних

Характеристики шахрайський транзакцій

summary	Amount	Class	Time	V11	V4	V2
count	492	492	492	492	492	492
mean	122.2113211382114	1.0	80746.80691056911	3.8001729113746077	4.542029104423093	3.623778101982281
stddev	256.6832882977121	0.0	47835.36513767505	2.678604522510197	2.8733176878992315	4.29121562613748
min	0.0	1	406.0	-1.70222840135659	-1.31327481447103	-8.40215367768915
max	2125.87	1	170348.0	12.0189131816199	12.1146718424589	22.0577289904909

Характеристики звичайних транзакцій

summary	Amount	Class	Time	V11	V4	V2
count	284315	284315	284315	284315	284315	284315
mean	88.29102242231271	0.0	94838.20225805884	-0.00657610422382...	-0.00785986782046...	-0.00627085741580...
stddev	250.10509222589235	0.0	47484.01578555089	1.003111906961151	1.3993332348712215	1.6361460525689606
min	0.0	0	0.0	-4.79747346479757	-5.68317119816995	-72.7157275629303
max	25691.16	0	172792.0	10.0021902173471	16.8753440335975	18.9024528401249



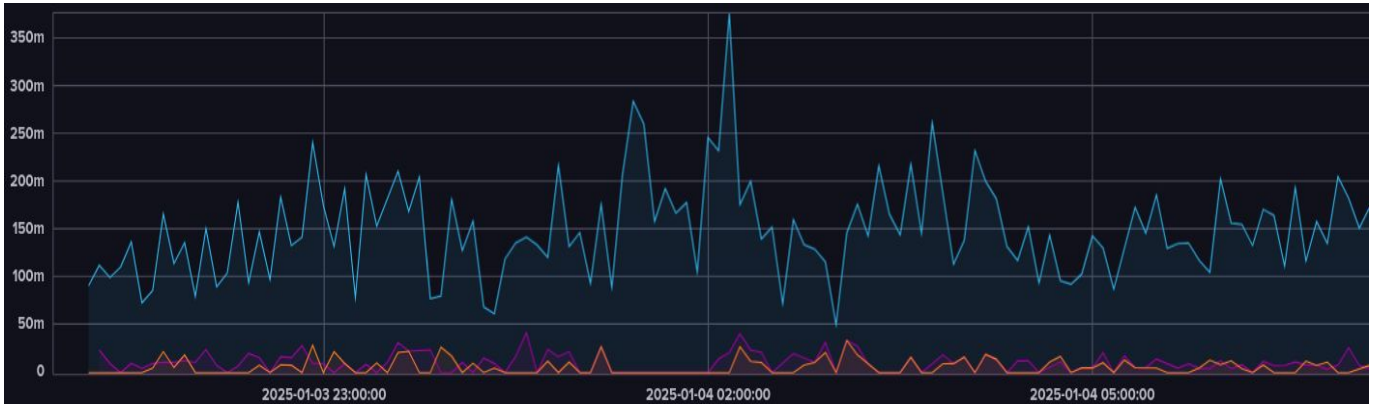
Архітектура системи

- Джерело даних: Потокове зчитування даних з файлів, розділених від основного набору.
- Обробка даних:
  - Попередня обробка: очищення та нормалізація.
  - Методи: статистичний аналіз (IQR), класифікація (Random Forest), виявлення аномалій (Isolation Forest).
- Зберігання результатів: Запис результатів у InfluxDB з тегами для ідентифікації методів виявлення шахрайства.

```
spark = SparkSession.builder \
    .appName("RealTimeAnomaliesDetection") \
    .master("spark://localhost:7077") \
    .config('spark.executor.memory', '15g') \
    .config('spark.driver.memory', '15g') \
    .config('spark.sql.shuffle.partitions', '200') \
    .getOrCreate()
```

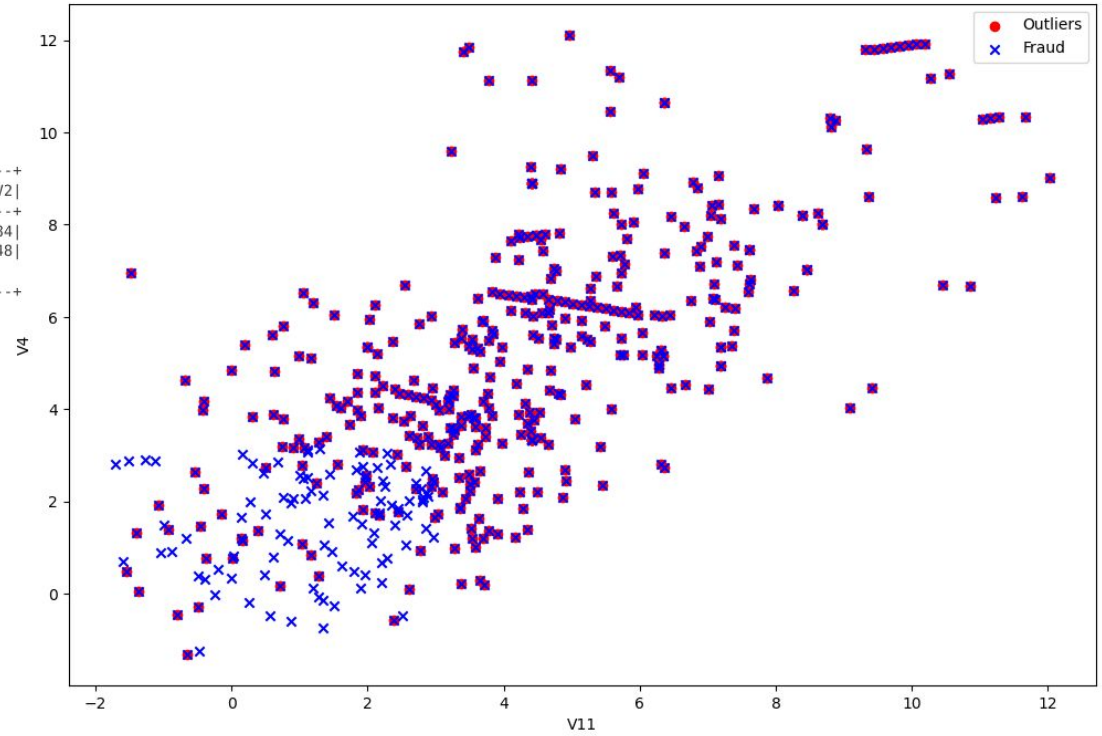
Візуалізація роботи методів для виявлення аномалій

- Синій графік - IQR
- Помаранчевий графік - RandomForest
- Фіолетовий графік - IsolationForest

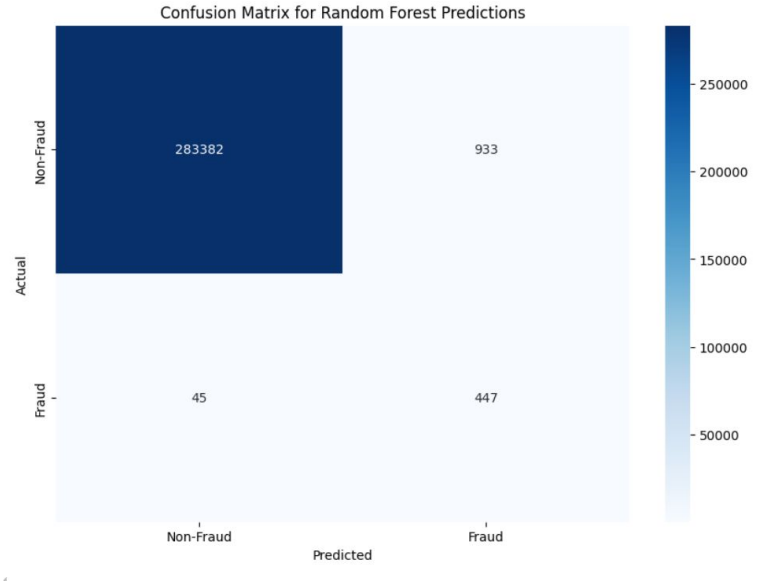


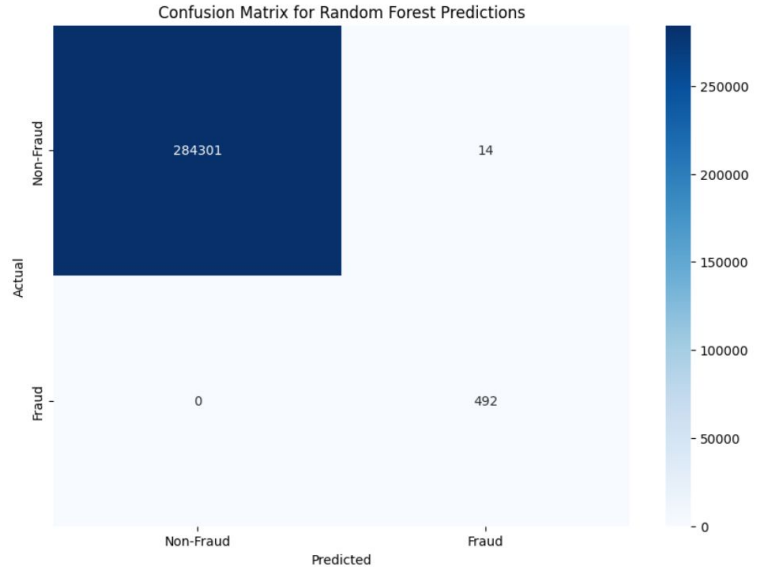
Візуалізація роботи статистичного методу IQR

```
+-------+-----------------+------------------+-----------------+-----------------+
|summary|           Amount|               V11|               V4|               V2|
+-------+-----------------+------------------+-----------------+-----------------+
|  count|            48284|             48284|            48284|            48284|
|   mean|349.0312353988893|-0.07078268497193412|0.5366772499739831|-0.892531682535448|
...
+-------+-----------------+------------------+-----------------+-----------------+
```



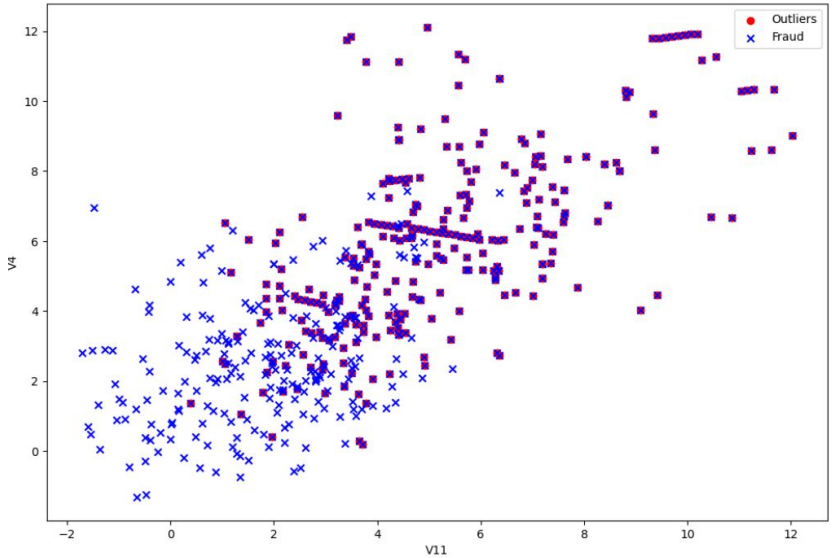
Матриця помилок для Random Forest





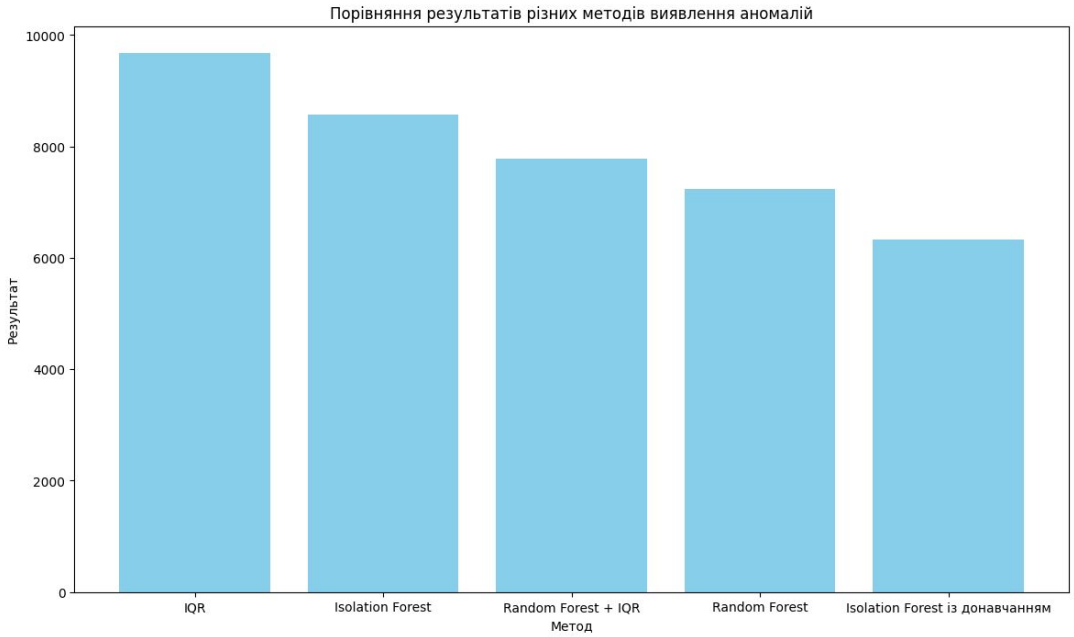
Візуалізація роботи методу Isolation Forest

	Amount	V11	V4	V2
count	2849.000000	2849.000000	2849.000000	2849.000000
mean	948.650232	0.766566	1.586276	-2.620376
std	1613.803997	2.371457	3.285752	9.881965
min	0.000000	-4.797473	-5.266509	-72.715728

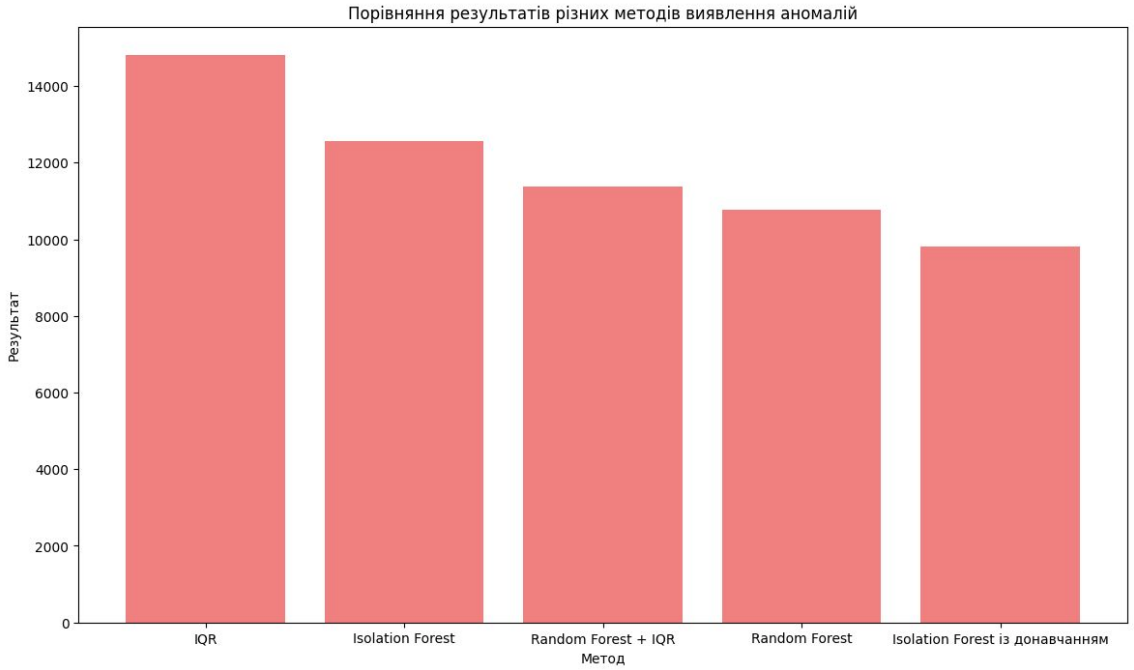


Тестування продуктивності

```
stream_data = spark.readStream \
    .format("csv") \
    .option("header", "true") \
    .option("maxFilesPerTrigger", 1) \
    .schema(data.schema) \
    .load("file:///opt/spark/data/stream_data")
query = stream_data.writeStream \
    .foreachBatch(process_batch) \
    .outputMode("append") \
    .trigger(processingTime="5 seconds") \
    .start()
```



```
query = stream_data.writeStream \
    .foreachBatch(process_batch) \
    .outputMode("append") \
    .start()
```



```
stream_data = spark.readStream \
    .format("csv") \
    .option("header", "true") \
    .schema(data.schema) \
    .load("file:///opt/spark/data/stream_data")
```