پروژه آنالیز بلادرنگ تراکنش‌های بانکی
و کشف تقلب

آرمین افضلی
محمدصالح پژند

فهرست مطالب

۱. مقدمه و شرح پروژه

۱.۱. هدف پروژه

هدف این پروژه پیاده‌سازی یک پایپلاین کلان‌داده (*Big Data Pipeline*) است که داده‌های واقعی تراکنش‌های بانکی را به صورت بلادرنگ پردازش کرده و ناهنجاری‌های زیر را شناسایی می‌کند:

- تراکنش‌های مشکوک از کاربران شناخته شده متقلب (*User_Hacker*)
- حملات بات (*Bot Attack*) با استفاده از تحلیل پنجره زمانی
- دسته‌بندی تراکنش‌ها بر اساس مبلغ (*Macro/Micro/Normal*)
- کاربران لیست سیاه با وضعیت *BLOCKED*

۲.۱. دیتاست

این پروژه از دیتاست *Credit Card Fraud Detection* از سایت *Kaggle* استفاده می‌کند:

۲. معماری سیستم

۱.۲. جریان داده

داده‌ها در این پایپلاین به صورت زیر جریان می‌یابند:

creditcard.csv → *Kafka Producer* → *Kafka Topic* → *Spark Streaming* → *Output*

۲.۲. اجزای سیستم

کامپوننت	توضیحات
Zookeeper	سرویس هماهنگ‌کننده کافکا
Apache Kafka	صف پیام توزیع‌شده با تاپیک *bank_transactions*
Kafka Producer	اسکریپت پایتون برای خواندن *CSV* و ارسال با شبیه‌سازی تقلب/بات
Spark Master	هماهنگ‌کننده کلاستر اسپارک
Spark Worker	اجراکننده تسک‌های پردازشی
Spark Processor	اپلیکیشن *Structured Streaming* با دسته‌بندی، لیست سیاه و تشخیص بات

جدول ۱: اجزای معماری سیستم

۳. گام اول: تولید جریان داده

۱.۳. پیاده‌سازی *Kafka Producer*

اسکریپت تولیدکننده فایل *creditcard.csv* را خط به خط خوانده و تراکنش‌ها را به کافکا ارسال می‌کند.

۲.۳. منطق تخصیص کاربر

سناریو	شرط	User_ID	عمل
تقلب	Class = 1	User_Hacker	ارسال یکبار
حمله بات	احتمال ۲٪	User_Bot	ارسال ۵ بار متوالی
عادی	پیش‌فرض	User_0001-1000	ارسال یکبار

جدول ۲: منطق تخصیص کاربر

۳.۳. کد پیاده‌سازی

```
if transaction_class == 1:
    user_id = 'User_Hacker'
elif random.random() < 0.02:
    user_id = 'User_Bot'
    for i in range(5):
        producer.send(topic, record)
else:
    user_id = random.choice(USER_POOL)
```

۴.۳. خروجی تولیدکننده (نتایج واقعی)

```
  Connected to Kafka at kafka:29092
  FRAUD DETECTED! Transaction TXN_1770283968053 - User: User_Hacker, Amount:
 $179.66
  BOT ATTACK SIMULATION - Sending 5 rapid transactions
  Bot attack completed - User: User_Bot, Amount: $18.13 x 5
  Processed 10000 transactions (Normal: 9054, Fraud: 36, Bot Attacks: 182)
```

۴. گام دوم: پردازش و تحلیل

۱.۴. دسته‌بندی تراکنش‌ها

یک ستون جدید به نام *Category* بر اساس مبلغ تراکنش ایجاد می‌شود:

دسته	شرط	توضیحات
Macro	*Amount* > $500	تراکنش‌های بزرگ
Micro	*Amount* < $20	تراکنش‌های کوچک
Normal	$20 < *Amount* < $500	تراکنش‌های معمولی

جدول ۳: دسته‌بندی تراکنش‌ها بر اساس مبلغ

```
categorized = stream.withColumn("Category",
    when(col("Amount") > 500, "Macro")
    .when(col("Amount") < 20, "Micro")
    .otherwise("Normal"))
```

۲.۴. تشخیص کاربران لیست سیاه (*JOIN*)

یک *DataFrame* دستی شامل کاربران لیست سیاه ایجاد شده و با استریم زنده *JOIN* می‌شود:

```
blacklist_data = [('User_Hacker', 'Known Fraud Actor', '2024-01-01')]
blacklist_df = spark.createDataFrame(blacklist_data, schema)

joined = transactions.join(blacklist_df, on='User_ID', how='left')

result = joined.withColumn('User_Status',
    when(col('Block_Reason').isNotNull(), 'BLOCKED')
    .otherwise('ACTIVE'))
```

۳.۴. تشخیص بات با پنجره زمانی (*Windowing*)

از قابلیت پنجره‌بندی اسپارک برای تشخیص الگوهای فرکانس بالا استفاده می‌شود:

- **مدت پنجره:** ۱۰ ثانیه (لغزان با فاصله ۵ ثانیه)
- **آستانه:** بیش از ۴ تراکنش در پنجره
- **هشدار:** *BOT_DETECTED*

```
windowed = stream.withWatermark('event_time', '20 seconds')
    .groupBy(
        window(col('event_time'), '10 seconds', '5 seconds'),
        col('User_ID')
    )
    .agg(count('*').alias('transaction_count'))
    .filter(col('transaction_count') > 4)
    .withColumn('Alert_Status', lit('BOT_DETECTED'))
```

۵. گام سوم: نمایش نتایج

۱.۵. خروجی تشخیص بات (نتایج واقعی)

```
Batch: 119
+-------------------+-------------------+--------+-----------------+------------+
|window_start       |window_end         |User_ID |transaction_count|Alert_Status|
+-------------------+-------------------+--------+-----------------+------------+
|2026-02-05 09:33:15|2026-02-05 09:33:25|User_Bot|5                |BOT_DETECTED|
|2026-02-05 09:33:10|2026-02-05 09:33:20|User_Bot|5                |BOT_DETECTED|
+-------------------+-------------------+--------+-----------------+------------+
```

۲.۵. آمار کلی (نتایج واقعی)

```
+-----------------+----------+----------+-----------+-------------+----------+-------
|total_transactions|macro_count|micro_count|normal_count|blocked_count|fraud_count|avg_amo
+-----------------+----------+----------+-----------+-------------+----------+-------
|175 |4 |80 |91 |1 |1 |68.33 |
+-----------------+----------+----------+-----------+-------------+----------+-------
```

۳.۵. آمار نهایی

معیار	مقدار
کل تراکنش‌های پردازش شده	*+10,300*
تراکنش‌های عادی	*9,327*
تراکنش‌های تقلبی (*User_Hacker*)	*38*
رویدادهای حمله بات	*187*
تراکنش‌های بات (*187 × 5*)	*935*

جدول ۴: آمار نهایی پردازش

۶. پیاده‌سازی *Docker*

۱.۶. سرویس‌های کانتینری

سرویس	ایمیج	کاربرد
zookeeper	*cp-zookeeper:7.5.0*	هماهنگی کافکا
kafka	*cp-kafka:7.5.0*	بروکر پیام
spark-master	*apache/spark:3.5.0*	هماهنگ‌کننده کلاستر
spark-worker	*apache/spark:3.5.0*	اجراکننده تسک
kafka-producer	*Custom Python*	استریم داده
spark-processor	*apache/spark:3.5.0*	پردازش استریم

جدول ۵: سرویس‌های کانتینری

۲.۶. دستورات اجرا

```
podman□compose up □d □□build

podman□compose logs □f kafka□producer

podman□compose logs □f spark□processor

podman□compose down □v
```