

Apache Kafka – Spark Integration

Using PySpark

Table of Contents

ASSIGNMENT II APACHE KAFKA – SPARK INTEGRATION	1
GENERAL ASSUMPTIONS:	3
ARCHITECTURE	3
EXERCISE 1.....	3
<i>Steps and Results for Exercise 1(a): Installing Kafka.....</i>	<i>3</i>
<i>Steps for Exercise 1(b): Send data from Excel to Kafka</i>	<i>5</i>
<i>For Exercise 1(c): For reading of messages from Kafka Topics and Processing in Spark.....</i>	<i>6</i>
<i>Steps and Results for Exercise 1(d): Create a Kafka topic and sending all the data into a topic.....</i>	<i>7</i>
EXERCISE 2: READ DATASET AND DEVELOP A MACHINE LEARNING MODEL TO DETECT THE CUSTOMER IS GOOD OR BAD	10
<i>1. Random forest classifier:</i>	<i>11</i>
<i>2. Extreme gradient boosted (XGBoost) classifier:.....</i>	<i>12</i>
EXERCISE 3: STREAM DATA PROCESSING FROM KAFKA AND SENDING ALERTS TO ANOTHER KAFKA	13
<i>Reading messages in Kafka</i>	<i>13</i>
<i>Pre-processing data after reading from Kafka.....</i>	<i>13</i>
<i>Processing Kafka data in Python with Machine Learning Model</i>	<i>15</i>
<i>Sending Classified data for Kafka Alert Topic</i>	<i>17</i>

General Assumptions:

- Customer transaction format assumed was same as the Kaggle dataset specified.
- There is no need for separate join with Master data since Customer master data was not available for the given dataset.
- There is no need to do SMS integration since it is out of scope for this course.
- Final fraud detection data and non-fraud data will be segregated and sent to different Kafka topic for consumption of application to alert.

Architecture

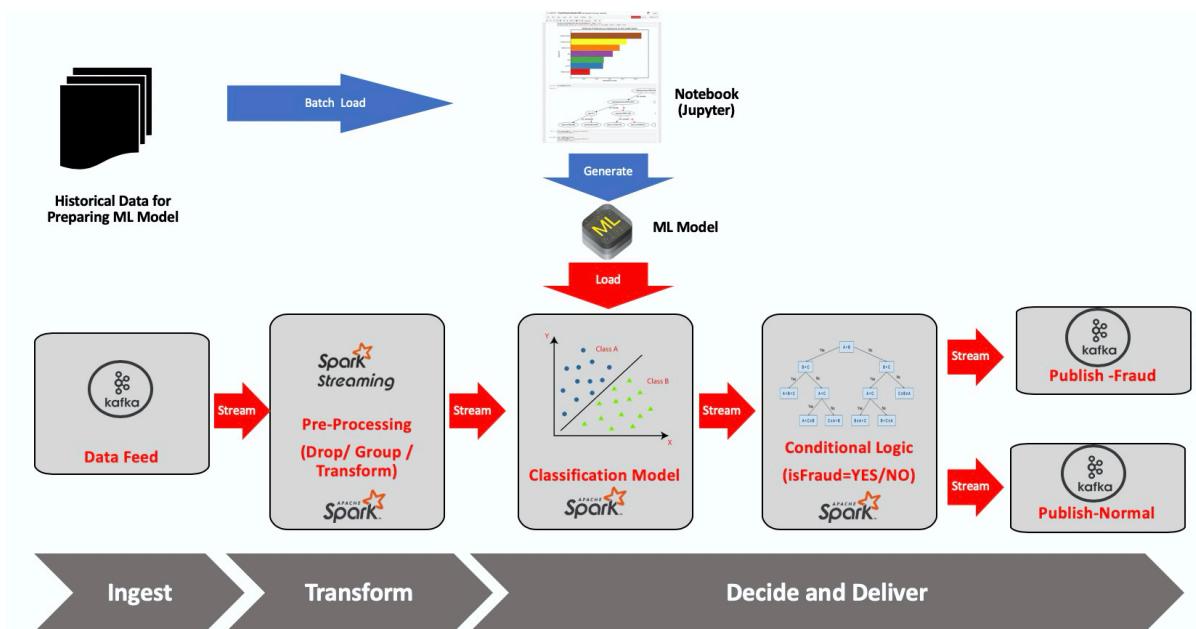


Fig 1: Architecture Diagram

Exercise 1

Scope:

- Install & Setup Kafka,
- Send and Store all the transaction data in Kafka from an Excel file using Python
- Read data from a Kafka topic using Python
- Install, Setup PySpark and test its functioniong

Steps and Results for Exercise 1(a): Installing Kafka

1. Download and Install Kafka as mentioned here
<https://kafka.apache.org/quickstart>
2. `cd ${KAFKA_HOME}`
3. Start Zookeeper (in a separate console)
`> bin/zookeeper-server-start.sh config/zookeeper.properties`
Result:

```
[2020-06-10 18:24:50,011] INFO Got user-level KeeperException when processing sessionid:0x1000685bd4f0000 type:create cxid:0x9 zxid:0xa txntype:-1 reqpath:/admin Error Path:/admin Error:KeeperErrorCode = NoNode for /admin (org.apache.zookeeper.server.PreReuestProcessor)
[2020-06-10 18:24:50,308] INFO Got user-level KeeperException when processing sessionid:0x1000685bd4f0000 type:create cxid:0x15 zxid:0x15 txntype:-1 reqpath:/cluster Error:KeeperErrorCode = NoNode for /cluster (org.apache.zookeeper.server.PreReuestProcessor)
[2020-06-10 18:24:51,254] INFO Got user-level KeeperException when processing sessionid:0x1000685bd4f0000 type:multi cxid:0x38 zxid:0x1c txntype:-1 reqpath:/admin/preferred_replica_election (org.apache.zookeeper.server.PreReuestProcessor)
[2020-06-10 18:25:41,464] INFO Got user-level KeeperException when processing sessionid:0x1000685bd4f0000 type:setData cxid:0x3e zxid:0x1d txntype:-1 reqpath:/config/topics/test Error:KeeperErrorCode = NoNode for /config/topics/test (org.apache.zookeeper.server.PreReuestProcessor)
[2020-06-10 18:28:09,094] INFO Got user-level KeeperException when processing sessionid:0x1000685bd4f0000 type:setData cxid:0x4b zxid:0x23 txntype:-1 reqpath:/config/topics/_consumer_offsets Error:KeeperErrorCode = NoNode for /config/topics/_consumer_offsets (org.apache.zookeeper.server.PreReuestProcessor)
```

Fig 2: Starting Zookeeper

4. Start Kafka Server (in a separate console)

>bin/kafka-server-start.sh config/server.properties

Result:

```
[2020-06-10 18:24:50,011] INFO Got user-level KeeperException when processing sessionid:0x1000685bd4f0000 type:create cxid:0x9 zxid:0xa txntype:-1 reqpath:/admin Error:KeeperErrorCode = NoNode for /admin (org.apache.zookeeper.server.PreReuestProcessor)
[2020-06-10 18:24:50,308] INFO Got user-level KeeperException when processing sessionid:0x1000685bd4f0000 type:create cxid:0x15 zxid:0x15 txntype:-1 reqpath:/cluster Error:KeeperErrorCode = NoNode for /cluster (org.apache.zookeeper.server.PreReuestProcessor)
[2020-06-10 18:24:51,254] INFO Got user-level KeeperException when processing sessionid:0x1000685bd4f0000 type:multi cxid:0x38 zxid:0x1c txntype:-1 reqpath:/admin/preferred_replica_election (org.apache.zookeeper.server.PreReuestProcessor)
[2020-06-10 18:25:41,464] INFO Got user-level KeeperException when processing sessionid:0x1000685bd4f0000 type:setData cxid:0x3e zxid:0x1d txntype:-1 reqpath:/config/topics/test Error:KeeperErrorCode = NoNode for /config/topics/test (org.apache.zookeeper.server.PreReuestProcessor)
[2020-06-10 18:28:09,094] INFO Got user-level KeeperException when processing sessionid:0x1000685bd4f0000 type:setData cxid:0x4b zxid:0x23 txntype:-1 reqpath:/config/topics/_consumer_offsets Error:KeeperErrorCode = NoNode for /config/topics/_consumer_offsets (org.apache.zookeeper.server.PreReuestProcessor)
```

Fig 3: Starting Kafka Server

5. Create a Topic (in a separate console)

>bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic test

6. Check the Topic

>bin/kafka-topics.sh --list --broker-list localhost:9092

Note: --bootstrap-server did not work on earlier versions, so used -brokerlist

7. Send messages to the created topic

>bin/kafka-console-producer.sh --bootstrap-server localhost:9092 --topic test

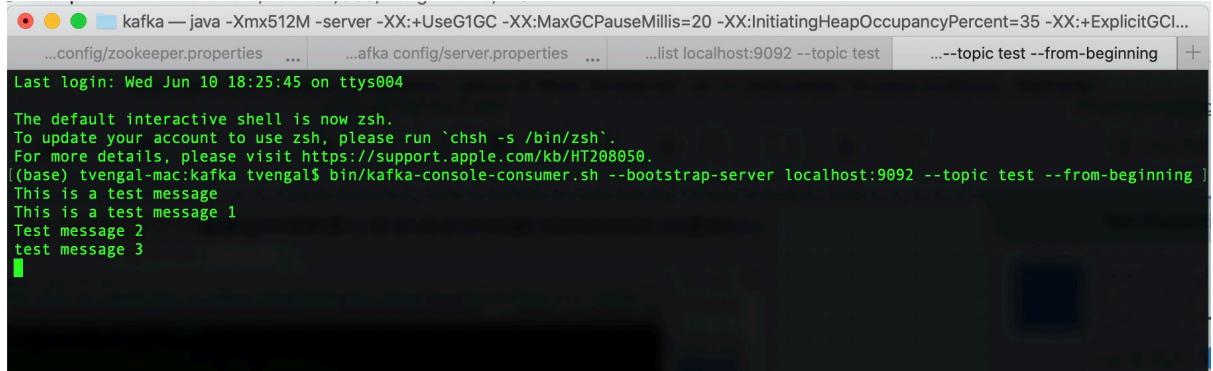
Result for steps 5,6,&7:

```
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) tvengal-mac:kafka tvengal$ bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic test
(base) tvengal-mac:kafka tvengal$ bin/kafka-topics.sh --list --bootstrap-server localhost:9092
(base) tvengal-mac:kafka tvengal$ bin/kafka-console-producer.sh --bootstrap-server localhost:9092 --topic test
This is a test message
>This is a test message 1
>Test message 2
>test message 3
>
```

Fig 4: Sending a test message in Kafka Producer

8. Start a Consumer and validate the topic

```
> bin/kafka-console-consumer.sh --bootstrap-server localhost:9092  
--topic test --from-beginning  
Result:
```



The screenshot shows a terminal window with several tabs at the top: 'kafka — java -Xmx512M -server -XX:+UseG1GC -XX:MaxGCPauseMillis=20 -XX:InitiatingHeapOccupancyPercent=35 -XX:+ExplicitGC...', '...config/zookeeper.properties ...', '...afka config/server.properties ...', '...list localhost:9092 --topic test', and '....topic test --from-beginning'. The main pane displays the following text:

```
Last login: Wed Jun 10 18:25:45 on ttys004  
  
The default interactive shell is now zsh.  
To update your account to use zsh, please run `chsh -s /bin/zsh`.  
For more details, please visit https://support.apple.com/kb/HT208050.  
[(base) tvengal-mac:kafka tvengal$ bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-beginning ]  
This is a test message  
This is a test message 1  
Test message 2  
test message 3
```

Fig 4: Receiving messages in Kafka Consumer client

Steps for Exercise 1(b): Send data from Excel to Kafka

- Import Python-Kafka library using the following command
>pip install kafka-python
- Run the program as given in the <fill>.ipynb
- This has 3 ways of sending
 1. Generate numeric messages and send in a loop
 2. Sending in Json format without header
 3. Sending in Json format with headers
- We chose to use the option (3) sending in Json format with headers

For Exercise 1(c): For reading of messages from Kafka Topics and Processing in Spark

1. Downloaded data set from <https://www.kaggle.com/ntnu-testimon/paysim1> ([Links to an external site.](#))
2. Tried the following options
 - a. Downloaded Hortonworks VM from Cloudera website, which was over 20GB. But could not install since it required more than 100GB of free space on laptop.
 - b. Looked for an alternate method to install manually Hadoop and Spark, but was had issues on Macbook.
 - c. Installed pyspark using
>pip install pyspark
Pyspark installation worked and this method was chosen to proceed.
3. Set the following environment variables
(Code: run bash_profile.txt)
 - a. PATH
 - b. CLASSPATH
 - c. SPARK_HOME
 - d. PYSPARK_DRIVER_PYTHON=jupyter
 - e. PYSPARK_DRIVER_PYTHON_OPTS='notebook'
4. Download **Spark-Streaming-Kafka** library from
5. Run the following command
>pyspark --master local[2]

Result:

```
Assignment2 — jupyter-notebook • java — 128x34
~/SA_Assign_2_Result_Summary-v1.docx
(base) tvengal-mac:Assignment2 tvengal$ chmod +x bash_profile.txt
(base) tvengal-mac:Assignment2 tvengal$ . bash_profile.txt
Successfully Ran Path Setting
(base) tvengal-mac:Assignment2 tvengal$ pyspark --master local[2]
[I 18:46:51.915 NotebookApp] The port 8888 is already in use, trying another port.
[I 18:46:51.916 NotebookApp] The port 8889 is already in use, trying another port.
[I 18:46:51.917 NotebookApp] The port 8890 is already in use, trying another port.
[I 18:46:52.157 NotebookApp] JupyterLab extension loaded from /Users/tvengal/opt/anaconda3/lib/python3.7/site-packages/jupyterlab
[b]
[I 18:46:52.157 NotebookApp] JupyterLab application directory is /Users/tvengal/opt/anaconda3/share/jupyter/lab
[I 18:46:52.160 NotebookApp] Serving notebooks from local directory: /Users/tvengal/Google Drive/BITS-DSE/3-StreamProcessingAnalytics/Assignment2
[I 18:46:52.160 NotebookApp] The Jupyter Notebook is running at:
[I 18:46:52.160 NotebookApp] http://localhost:8891/?token=776b29cb4a11687dea732cdf5855295617d71165ba268ee6
[I 18:46:52.160 NotebookApp] or http://127.0.0.1:8891/?token=776b29cb4a11687dea732cdf5855295617d71165ba268ee6
[I 18:46:52.160 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 18:46:52.176 NotebookApp]

To access the notebook, open this file in a browser:
  file:///Users/tvengal/Library/Jupyter/runtime/nbserver-58683-open.html
Or copy and paste one of these URLs:
  http://localhost:8891/?token=776b29cb4a11687dea732cdf5855295617d71165ba268ee6
  or http://127.0.0.1:8891/?token=776b29cb4a11687dea732cdf5855295617d71165ba268ee6
[W 18:47:10.477 NotebookApp] Notebook Arti_notebook.ipynb is not trusted
[I 18:47:11.191 NotebookApp] Kernel started: edff99ef-481e-40c3-92c2-a5a8b79e1c28
20/06/10 18:47:18 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
[I 18:49:10.932 NotebookApp] Saving file at /Arti_notebook.ipynb
[W 18:49:10.932 NotebookApp] Notebook Arti_notebook.ipynb is not trusted
```

Fig 4: Starting Pyspark

6. This will open up the Jupyter Notebook
7. In a Python Notebook and run the following commands
 - a. SC

This will display the SparkContext

b. `os.environ['PYSPARK_SUBMIT_ARGS'] = '--jars {SPARKHOME} /jars/spark-streaming-kafka-0-8_2.11-2.0.2.jar pyspark-shell'`

This will load the spark-streaming-kafka library to PYSPARK and other libraries as posted.

Result:

```
In [1]: sc
Out[1]: SparkContext
          Spark UI
          Version
          v2.4.5
          Master
          local[2]
         AppName
          PySparkShell

In [2]: import findspark
findspark.init()

import pyspark
import random
sc = SparkContext.getOrCreate()
#sc = pyspark.SparkContext(appName="Pi")
num_samples = 1000

def inside(p):
    x, y = random.random(), random.random()
    return x*x + y*y < 1

count = sc.parallelize(range(0, num_samples)).filter(inside).count()

pi = 4 * count / num_samples
print(pi)

sc.stop()

3.072
```

Fig 5: Spark execution on Python Jupyter Notebook

8. Rest of the code is available in the “ReadKafka_RunSpark.ipynb” notebook

Steps and Results for Exercise 1(d): Create a Kafka topic and sending all the data into a topic

- Code available at “Send-CSV-to-Kafka-v2.ipynb”
- This code has 3 different options to send to Kafka
- a) For simple test of sending messages to Kafka in format {"number": #} every second
 - b) For sending a complete CSV file in JSON without header
 - c) For sending a complete CSV file in JSON with first line as header

```

In [1]: from time import sleep
import json
from json import dumps
from kafka import KafkaProducer
import csv
import pandas
import yaml

In [2]: producer = KafkaProducer(bootstrap_servers=['localhost:9092'], value_serializer=lambda x: dumps(x).encode('utf-8'))

In [5]: #a) for simple test of sending messages to Kafka in format {"number": #} every second
for e in range(100):
    data = {'number' : e}
    producer.send('numtest', value=data)
    sleep(1)

In [6]: #b) for sending a complete CSV file in JSON without header
with open('../PS_log-test.csv') as f:
    for line in f:
        row = yaml.safe_load(line)
        jd = json.dumps(row)
        producer.send('csvtopic',jd) #topic name , data

In [8]: #c) for sending a complete CSV file in JSON with first line as header
from kafka import KafkaProducer
import time,csv

# create producer to kafka connection
producer = KafkaProducer(bootstrap_servers='localhost:9092')
# define *.csv file and a char that divide value
fname = "../PS_log-test.csv"
divider_char = ','
# open file
with open(fname) as fp:
    # read header (first line of the input file)
    line = fp.readline()
    header = line.split(divider_char)

    #loop other data rows
    line = fp.readline()
    while line:
        # start to prepare data row to send
        data_to_send = ""
        values = line.split(divider_char)
        len_header = len(header)
        for i in range(len_header):
            data_to_send += "\\" + header[i].strip() + ":" + values[i].strip() + "\\"
            if i<len_header-1 :
                data_to_send += ","
        data_to_send = "{" + data_to_send + "}"

        # send data via producer
        producer.send('test', bytes(data_to_send, encoding='utf-8'))
        line = fp.readline()
        # A это max)) на всякий случай
        #time.sleep(1)
producer.close()

```

Fig:6 Jupyter notebook with 3 different options to send Kafka messages

```

{"number": 71}
{"number": 72}
{"number": 73}
{"number": 74}
{"number": 75}
{"number": 76}
{"number": 77}
{"number": 78}
{"number": 79}
{"number": 80}
{"number": 81}
{"number": 82}
{"number": 83}
{"number": 84}
{"number": 85}
{"number": 86}
{"number": 87}
{"number": 88}
{"number": 89}
{"number": 90}
{"number": 91}
{"number": 92}
{"number": 93}
{"number": 94}
{"number": 95}
{"number": 96}
{"number": 97}
{"number": 98}
{"number": 99}

```

Fig 7(a) Result of sending option (a)

```
(base) tvengal-mac:kafka tvengal$ bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-beginning
This is a test message
This is a test message 1
Test message 2
test message 3
^CProcessed a total of 4 messages
(base) tvengal-mac:kafka tvengal$ bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test1 --from-beginning

"\\"step.type,amount.nameOrig,oldbalanceOrg,newbalanceOrig.nameDest,oldbalanceDest,newbalanceDest,isFraud,isFlaggedFraud\""
"\\"1,PAYMENT,9839.64,C1231006815,170136.0,160296.36,M1979787155.0,0.0,0.0,0.0\\""
"\\"1,PAYMENT,1864.28,C1666544295,21249.0,19384.72,M2044282225.0,0.0,0.0,0.0\\""
"\\"1,TRANSFER,181.0,C1305486145,181.0,0.0,C553264065.0,0.0,0.1,0\\""
"\\"1,CASH_OUT,181.0,C8400083671,181.0,0.0,C38997010,21182.0,0.0,1,0\\""
"\\"1,PAYMENT,11668.14,C2048537720,41554.0,29885.86,M1230701703.0,0.0,0.0,0.0\\""
"\\"1,PAYMENT,7817.71,C90045638,53860.0,46042.29,M573487274.0,0.0,0.0,0.0\\""
"\\"1,PAYMENT,7107.77,C154988899,183195.0,176087.23,M408069119.0,0.0,0.0,0.0\\""
"\\"1,PAYMENT,7861.64,C1912850431,176087.23,168225.59,M633326333.0,0.0,0.0,0.0\\""
"\\"1,PAYMENT,4024.36,C1265012928,2671.0,0.0,M176932104.0,0.0,0.0,0.0\\""
"\\"1,DEBIT,5337.77,C712410124,41720.0,36382.23,C1956008860,41898.0,40348.79,0,0\\""
"\\"1,DEBIT,9644.94,C1900366749,4465.0,0.0,C997608398,10845.0,157982.12,0,0\\""
"\\"1,PAYMENT,3099.97,C249177573,20771.03,M2096539129.0,0.0,0.0,0.0\\""
"\\"1,PAYMENT,2560.74,C1648232591,5070.0,2509.26,M972865270.0,0.0,0.0,0.0\\""
"\\"1,PAYMENT,11633.76,C1716932897,10127.0,0.0,M801569151.0,0.0,0.0,0.0\\""
"\\"1,PAYMENT,4098.78,C1026483832,503264.0,499165.22,M1635378213.0,0.0,0.0,0.0\\""
```

Fig 7(b) : Result of sending option (b)

```
{"step": "1", "type": "PAYMENT", "amount": "25.12", "nameOrig": "C1257299717", "oldbalanceDest": "0", "newbalanceDest": "0"}, {"step": "1", "type": "PAYMENT", "amount": "17150.89", "nameOrig": "C181572244", "oldbalanceDest": "0", "newbalanceDest": "0"}, {"step": "1", "type": "PAYMENT", "amount": "54486.99", "nameOrig": "M1733022752", "oldbalanceDest": "0", "newbalanceDest": "0"}, {"step": "1", "type": "TRANSFER", "amount": "679502.24", "nameOrig": "C722417467", "oldbalanceDest": "290", "newbalanceDest": "0"}, {"step": "1", "type": "TRANSFER", "amount": "177652.91", "nameOrig": "C753631393", "oldbalanceDest": "23720", "newbalanceDest": "0"}, {"step": "1", "type": "PAYMENT", "amount": "5443.26", "nameOrig": "C1262869688", "oldbalanceDest": "0", "newbalanceDest": "0"}, {"step": "1", "type": "PAYMENT", "amount": "1175.59", "nameOrig": "C1166106620", "oldbalanceDest": "0", "newbalanceDest": "0"}, {"step": "1", "type": "PAYMENT", "amount": "1097.74", "nameOrig": "C221861886", "oldbalanceDest": "257978.49", "newbalanceDest": "0"}, {"step": "1", "type": "PAYMENT", "amount": "8281.57", "nameOrig": "C900298796", "oldbalanceDest": "257733.62", "newbalanceDest": "0"}, {"step": "1", "type": "PAYMENT", "amount": "1175.29", "nameOrig": "C603658030", "oldbalanceDest": "20509", "newbalanceDest": "0"}, {"step": "1", "type": "PAYMENT", "amount": "1097.74", "nameOrig": "C221861886", "oldbalanceDest": "24866", "newbalanceDest": "0"}, {"step": "1", "type": "TRANSFER", "amount": "184986.8", "nameOrig": "C697508322", "oldbalanceDest": "39588", "newbalanceDest": "0"}, {"step": "1", "type": "PAYMENT", "amount": "3671.32", "nameOrig": "C361380654", "oldbalanceDest": "100094", "newbalanceDest": "0"}, {"step": "1", "type": "DEBIT", "amount": "3580.26", "nameOrig": "C1579132337", "oldbalanceDest": "2565", "newbalanceDest": "0"}, {"step": "1", "type": "PAYMENT", "amount": "8550.9", "nameOrig": "C1795225096", "oldbalanceDest": "40060", "newbalanceDest": "0"}, {"step": "1", "type": "PAYMENT", "amount": "31509.1", "nameOrig": "M790094605", "oldbalanceDest": "0", "newbalanceDest": "0"}]
```

Fig 7(c) : Result of sending option (c)

Exercise 2: Read Dataset and develop a machine learning model to detect the customer is good or bad

Scope

- Build a Machine learning model with good accuracy from the given static data.
- This built model should be usable in a stream-processing framework and should be fast enough to detect fraudulent activity in the account.

Summary:

Paysim synthetic dataset of mobile mobile money transactions. Each step represents an hour of simulation. The features present in the dataset are: step, type, amount, nameOrig, nameDest, oldbalanceOrig, newbalanceOrig, nameDest, oldbalanceDest, newbalanceDest, isFraud. As part of our analysis we are dropping 'nameOrig'(customer who started the transaction), 'nameDest'(recipient ID of the transaction) because fraudulent transactions are not indicated by these features.

Steps:

1. Import Data: Import dataset to a Python Dataframe
2. Exploratory Data Analysis
 - i. List the data, and preview the data and check whether any null values are there

```
In [4]: #List the columns
list(df)

Out[4]: ['step',
 'type',
 'amount',
 'nameOrig',
 'oldbalanceOrg',
 'newbalanceOrig',
 'nameDest',
 'oldbalanceDest',
 'newbalanceDest',
 'isFraud',
 'isFlaggedFraud']

In [5]: print(df.head())

      step      type    amount   nameOrig  oldbalanceOrg  newbalanceOrig \
0       1  PAYMENT  9839.64  C1231006815     170136.0     160296.36
1       1  PAYMENT  1864.28  C1666544295     21249.0      19384.72
2       1  TRANSFER  181.00  C1305486145      181.0        0.00
3       1  CASH_OUT  181.00  C840083671      181.0        0.00
4       1  PAYMENT  11668.14  C2048537720     41554.0      29885.86

           nameDest  oldbalanceDest  newbalanceDest  isFraud  isFlaggedFraud
0  M1979787155        0.0          0.0         0         0
1  M2044282225        0.0          0.0         0         0
2  C553264065         0.0          0.0         1         0
3  C38997010        21182.0        0.0         1         0
4  M1230701703        0.0          0.0         0         0

Test if there any missing values in DataFrame. It turns out there are no obvious missing values but, as we will see below, this does not rule out proxies by a numerical value like 0.

In [36]: df.isnull().values.any()

Out[36]: False
```

Fig 8: Reading the data in Python

ii. List the details of the dataset

1. Total number of Records = 6362620
2. Total number of Records with Fraudulent Transactions = 8213
3. Total number of Records with Flagged Fraudulent Transactions = 16
4. Percentage Ratio of Fraudulent Transactions= 0.12908204481801522
5. Total number of Records with type PAYMENT = 2151495
6. Total number of Records with type TRANSFER = 532909

7. Total number of Records with type CASH_IN = 1399284
8. Total number of Records with type CASH_OUT = 2237500
9. Total number of Records with type DEBIT = 41432

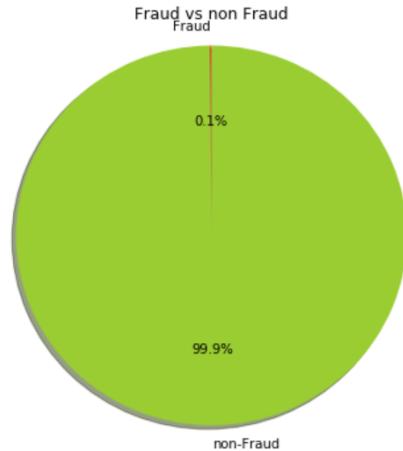


Fig 9 : Pie chart on the Fraud vs Non-Fraud

The feature 'isFlaggedFraud'(flagging a fraudulent transaction (1) and non-fraudulent (0)) is dropped. Because it is not set even in the scenarios of fraudulent transactions.
The target label used here is isFraud which classifies the transactions as fraudulent or not. From our dataset it is observed that 99.9% of the transactions are non-fraudulent and 0.1% are fraudulent as shown below:

iii. Identify the Fraudulent transaction types

- Fraudulent transactions types are ['TRANSFER', 'CASH_OUT']

iv. Count the number of transactions in each of the category

- Total number of Records with fraudulent TRANSFERS = 4097
- Total number of Records with fraudulent CASH_OUTs = 4116
- Total number of Records with isFlaggedfraudulent with TRANSFERS = 16
- Total number of Records with isFlaggedfraudulent with CASH_OUTs = 0

3. Build Machine Learning Model:

We propose 2 classifiers best suitable for the given problem statement and their details are mentioned as below:

- i. Two different models were built, evaluated and used
 1. Random Forest based Classifier
 2. XGBoost based Classifier

Following are the details about both the Machine learning Models:

1. Random forest classifier:

The random forest is a classification algorithm consisting of many decision trees. It uses bagging and features randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

In the proposed model, the pre-processing is performed on the dataset where the feature ‘type’ is converted into categorical data. Also the features like ‘nameOrig’, ‘nameDest’ and ‘isFlaggedFraud’ have been dropped from the dataset. The dataset is split in the ratio of 80:20 such that 80% of the records are used as training data and 20% of the records are used as testing data. The Random forest classifier has been built with n_estimators(number of estimators) as 100 and maximum depth as 10. The evaluation metrics used are ‘roc_auc_score’ which returned a score of 86% and confusion_matrix.

The trained model is then saved to a pickle file and can be loaded for further can be loaded for further stream processing deployment.

Code Available at “FraudPredictionModel-RandomForest.ipynb”

2. Extreme gradient boosted (XGBoost) classifier:

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. XGBoost dominates structured or tabular datasets on classification and regression predictive modeling problems. This algorithm goes by lots of different names such as gradient boosting, multiple additive regression trees, stochastic gradient boosting or gradient boosting machines. Boosting is an ensemble technique where new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made.

The pre-processing is performed on the dataset where the feature ‘type’ is converted into categorical data. Also the features like ‘nameOrig’, ‘nameDest’ and ‘isFlaggedFraud’ have been dropped from the dataset. The dataset is split in the ratio of 80:20 such that 80% of the records are used as training data and 20% of the records are used as testing data. The XGBoost classifier is built with a maximum depth of 3 and n_jobs(number of parallel threads) as 4. The evaluation metric used here is Area under the precision-recall curve (AUPRC) which returned 94.3%.

The trained model is then saved to a binary file and can be loaded for further stream processing deployment.

Code Available at “FraudPredictionModel-XGB.ipynb”

Exercise 3: Stream Data Processing from Kafka and Sending Alerts to another Kafka

Scope

- Read messages from Kafka
- Pre-process the incoming messages in near real-time
- Import and Run Machine Learning model in near real-time
- Filter out the fraudulent transaction data
- Join the fraudulent transaction data with the original dataset
- Send the fraudulent transaction data to a designated Kafka topic and send the non-fraudulent transaction data to another Kafka topic

Reading messages in Kafka

Code available at FraudDetection.py

- a. using Python program
using Streaming Context and Create Stream from *readStream* from Kafka
- b. Read data is parsed in json format and stored into another alias

```
from kafka import KafkaProducer
import json

conf = SparkConf().setAppName("FraudDetection").setMaster("local").set("spark.io.compression.codec", "snappy")
sc = SparkContext(conf=conf)
sc.setLogLevel("ERROR")
spark = SparkSession(sc)
sqlContext = SQLContext(sc)

jsonschema = StructType().add("step", StringType()) \
    .add("type", StringType()) \
    .add("amount", StringType()) \
    .add("nameOrig", StringType()) \
    .add("oldbalanceOrg", StringType()) \
    .add("newbalanceOrg", StringType()) \
    .add("nameDest", StringType()) \
    .add("oldbalanceDest", StringType()) \
    .add("newbalanceDest", StringType())

df = spark.readStream.format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe", "test") \
    .option("startingOffsets", "earliest").load() \
    .select(from_json(col("value").cast("string"), jsonschema).alias("parsed_mta_values"))
```

Fig 10: Code displaying using Spark Context for reading data from

Pre-processing data after reading from Kafka

1. The standard JSON frame was not used in the next stage a full JSON sending code was used
2. The streaming data was read with a defined Json Schema into a pyspark.sql.dataframe and then converted to the Panda Dataframes for pre-processing

```

print("raw Type", type(raw))
print("raw ", raw)
raw.show()

mta_data = raw.toPandas()
org_mta_data = mta_data
print("mta_data", mta_data)

```

Fig 11: Code for transforming dataframe to Pandas data frame

...nalytics/Assignment2 — bash							...— jupyter-notebook + javapython3 notebook + javassignment2 — java + python +			
20/06/14 12:31:31 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, 192.168.0.104, 55099, None)														
20/06/14 12:31:31 INFO BlockManagerEndpoint: Registering block manager 192.168.0.104:55099 with 366.3 MB RAM, BlockManagerId(driver, 192.168.0.104, 55099, None)														
20/06/14 12:31:31 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, 192.168.0.104, 55099, None)														
20/06/14 12:31:31 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, 192.168.0.104, 55099, None)														
root														
-- parsed_mta_values: struct (nullable = true)														
-- step: string (nullable = true)														
-- type: string (nullable = true)														
-- amount: string (nullable = true)														
-- nameOrig: string (nullable = true)														
-- oldbalanceOrg: string (nullable = true)														
-- newbalanceOrg: string (nullable = true)														
-- nameDest: string (nullable = true)														
-- oldbalanceDest: string (nullable = true)														
-- newbalanceDest: string (nullable = true)														
Mta_data Type <class 'pyspark.sql.dataframe.DataFrame'>														
raw Type <class 'pyspark.sql.dataframe.DataFrame'>														
raw DataFrame[step: string, type: string, amount: string, nameOrig: string, oldbalanceOrg: string, newbalanceOrg: string, nameDest: string, oldbalanceDest: string, newbalanceDest: string]														
+---+-----+-----+-----+-----+-----+-----+-----+-----+-----+														
step type amount nameOrig oldbalanceOrg newbalanceOrg nameDest oldbalanceDest newbalanceDest														
+---+-----+-----+-----+-----+-----+-----+-----+-----+-----+														
90 CASH_OUT 7696836.35 C1784092244 7696836.35 0 C646123724 180993.45 7877829.8														
90 TRANSFER 39664.74 C1509358186 39664.74 0 C2068734277 0 0 0														
90 CASH_OUT 39664.74 C1334465444 39664.74 0 C247088283 0 0 39664.74														
90 TRANSFER 249791.3 C2018224474 249791.3 0 C1562137241 0 0 0														
90 CASH_OUT 249791.3 C712828337 249791.3 0 C1973032153 0 0 249791.3														
91 TRANSFER 921668.95 C1152872254 921668.95 0 C1484049545 0 0 0														
91 CASH_OUT 921668.95 C1505563519 921668.95 0 C863042157 96395.54 1018064.5														
91 TRANSFER 737285.03 C2082041963 737285.03 0 C484298628 0 0 0														
91 CASH_OUT 737285.03 C830817076 737285.03 0 C235145980 0 0 737285.03														
91 TRANSFER 28308.97 C531625667 28308.97 0 C1312031909 0 0 0														
91 CASH_OUT 28308.97 C1851254039 28308.97 0 C1151857075 0 0 28308.97														
91 TRANSFER 482679.6 C822183230 482679.6 0 C1393101702 0 0 0														
91 CASH_OUT 482679.6 C1126237876 482679.6 0 C842293503 0 0 482679.6														
92 TRANSFER 1461930.03 C593029629 1461930.03 0 C1743578484 0 0 0														
92 CASH_OUT 1461930.03 C1928781788 1461930.03 0 C1636281428 3762496.73 5224426.76														
92 TRANSFER 271908.4 C1340312777 271908.4 0 C863062485 0 0 0														
92 CASH_OUT 271908.4 C71055143 271908.4 0 C1732817669 1488088.26 1759996.66														
92 TRANSFER 101903.69 C1615802716 101903.69 0 C1977391812 0 0 0														
92 CASH_OUT 101903.69 C612571830 101903.69 0 C544677509 280117.63 382021.32														
92 TRANSFER 21574.55 C2002083627 21574.55 0 C1024161943 0 0 0														
+---+-----+-----+-----+-----+-----+-----+-----+-----+-----+														
only showing top 20 rows														

Fig 12: Data received over stream converted into specified Dataframe (pyspark.sql.dataframes type)

...nalytics/Assignment2 — bash									...— jupyter-notebook > javapython3 notebook > javassignment2 — java + python ...		
mta_data			step	type	amount	...	nameDest	oldbalanceDest	newbalanceDest		
0	90	CASH_OUT	7696836.35	...	C646123724	180993.45	7877829.8										
1	90	TRANSFER	39664.74	...	C2068734277	0	0										
2	90	CASH_OUT	39664.74	...	C247088283	0	39664.74										
3	90	TRANSFER	249791.3	...	C1562137241	0	0										
4	90	CASH_OUT	249791.3	...	C1973032153	0	249791.3										
...		
196	1	TRANSFER	184986.8	...	C1688019098	52340	97263.78										
197	1	PAYOUT	1718.29	...	M1689924104	0	0										
198	1	PAYOUT	3671.32	...	M631673932	0	0										
199	1	DEBIT	3580.26	...	C1321640594	21185	18910.85										
200	1	PAYOUT	8550.9	...	M790094605	0	0										
[201 rows x 9 columns]																	
New Mta data																	
0	90	0	7696836.35	...	0.00	180993.45	7877829.80										
1	90	3	39664.74	...	0.00	0.00	0.00										
2	90	0	39664.74	...	0.00	0.00	39664.74										
3	90	3	249791.30	...	0.00	0.00	0.00										
4	90	0	249791.30	...	0.00	0.00	249791.30										
...		
196	1	3	184986.80	...	0.00	52340.00	97263.78										
197	1	2	1718.29	...	116092.71	0.00	0.00										
198	1	2	3671.32	...	96422.68	0.00	0.00										
199	1	1	3580.26	...	0.00	21185.00	18910.85										
200	1	2	8550.90	...	31509.10	0.00	0.00										
[201 rows x 7 columns]																	

Fig 13: Data after pre-processing, dropping irrelevant columns and formatting data to numeric type

```
Note: Tried reading as DataFrame object, but then had an error
20/06/11 01:56:16 WARN KafkaOffsetReader: Error in attempt 1 getting K
afka offsets:
org.apache.kafka.common.config.ConfigException: Missing required config
uration "partition.assignment.strategy" which has no default value.
Found a solution to use Kafka 0.10.0.1 client libraries from
: https://stackoverflow.com/questions/44959483/pyspark-structured-streaming-kafka-configuration-error?rq=1
```

Processing Kafka data in Python with Machine Learning Model

- The processed dataframe is passed to Machine Learning Model that was derived from earlier step.
- Classification Result of the ML Model is then added back to the original data
- Only one of the Classifier model is required to be loaded into the Spark Context . We used RandomForest Classifier Model here. (XGB was not used since it required more time and effort, but it can also be used in similar manner)

```
with open('Classifier2.pickle', 'rb') as handle:
    RFmodel = pickle.load(handle)
RFmodel kevs()

RFclf = RFmodel['classifier']
predictions = RFclf.predict(mta data)
print (predictions)

ora mta data['isFraud']=predictions
new mta data=ora mta data
print("New Mta data with predictions". new mta data)
```

Fig 14: Code for loading Machine Learning Model and sending prepared dataset to Model and joining with original data

```
...nalytics/Assignment2 --bash ...— jupyter-notebook + java ... ...python3 notebook + java ... ...ssignment2 — java + python +  
/Users/tvengal/opt/anaconda3/lib/python3.7/site-packages/sklearn/utils/deprecation.py:144: FutureWarning: The sklearn.tree module is deprecated in version 0.22 and will be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.tree. Anything that cannot be imported from sklearn.tree is now part of the private API.  
    warnings.warn(message, FutureWarning)  
/Users/tvengal/opt/anaconda3/lib/python3.7/site-packages/sklearn/base.py:318: UserWarning: Trying to unpickle estimator DecisionTreeClassifier from version 0.21.3 when using version 0.22.1. This might lead to breaking code or invalid results. Use at your own risk.  
    UserWarning)  
/Users/tvengal/opt/anaconda3/lib/python3.7/site-packages/sklearn/base.py:318: UserWarning: Trying to unpickle estimator RandomForestClassifier from version 0.21.3 when using version 0.22.1. This might lead to breaking code or invalid results. Use at your own risk.  
    UserWarning)  
[1 0 0 0 1 1 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 1 1  
0 0 1 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
New Mta data with predictions      step type      amount ... oldbalanceDest newbalanceDest isFraud  
0   90     0  7696836.35 ...  180993.45    7877829.00     1  
1   90     3  39664.74 ...       0.00        0.00     0  
2   90     0  39664.74 ...       0.00    39664.74     0  
3   90     3  249791.30 ...       0.00        0.00     0  
4   90     0  249791.30 ...       0.00    249791.30     0  
...   ...   ...   ...   ...   ...   ...  
196    1     3  184986.80 ...  52340.00    97263.78     0  
197    1     2  1718.29 ...       0.00        0.00     0  
198    1     2  3671.32 ...       0.00        0.00     0  
199    1     1  3580.26 ...  21185.00   18910.85     0  
200    1     2  8550.90 ...       0.00        0.00     0  
[281 rows x 10 columns]
```

Fig 15: Data result after sending to Machine Learning Model and Merging with the original dataset

For running as a continuous program:

1. Modify the following line

```
raw_table= mta_data.writeStream.outputMode("append").queryName("raw_table").format("memory").start()
          to
raw_table= mta_data.writeStream.outputMode("append").queryName("raw_table").format("memory").start().awaitTermination()
```

2. Run the following command

```
spark-submit --class MainClass --jars /Users/tvengal/opt/anaconda3/lib/python3.7/site-packages/pyspark/jars/kafka-clients-0.10.0.1.jar --driver-class-path /Users/tvengal/opt/anaconda3/lib/python3.7/site-packages/pyspark/jars/kafka-clients-0.10.0.1.jar FraudDetection.py
```

Sending Classified data for Kafka Alert Topic

```
Fraud_prepared step          90
type                      0
amount        7.69684e+06
nameOrig      C1784092244
oldbalanceOrg  7.69684e+06
newbalanceOrig 0
nameDest       C646123724
oldbalanceDest 180993
newbalanceDest 7.87783e+06
isFraud        1
Name: 0, dtype: object
```

Fig 16: Prepared dataset ready to be sent to Kafka

```
# prepared record => X test which is fraud
for i in range(0, len(new_mta_data)):
    prepared_record=new_mta_data.iloc[i,:]
    if (prepared_record.isFraud==1) :
        print("\nFraud prepared", prepared_record)
        producer = KafkaProducer(bootstrap_servers='localhost:9092', value_serializer=lambda v: json.dumps(v).encode('utf-8'))
        producer.send("FraudData", value=prepared_record.to_json().dumps(timeout=30))
    else:
        print("\nNo Fraud prepared", prepared_record)
        producer = KafkaProducer(bootstrap_servers='localhost:9092', value_serializer=lambda v: json.dumps(v).encode('utf-8'))
        producer.send("NonFraudData", value=prepared_record.to_json().dumps(timeout=30))
```

Fig 17: Data result after sending to Machine Learning Model and Merging with the original dataset

```
...ig/zookeeper.properties ... ...config/server.properties ... ...udData --from-beginning ~/software/kafka -- bash | +
[(base) tvengal-mac:kafka tvengal$ bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic FraudData --from-beginning
"(\\"step\\":90,0,\\\"type\\\":0,0,\\\"amount\\\":7696836.349999996,\\\"oldbalanceOrg\\\":7696836.349999996,\\\"newbalanceOrig\\\":0,0,\\\"oldbalanceDest\\\":180993.45,\\\"newbalanceDest\\\":7877829.799999998,\\\"isFraud\\\":1.0)"
"(\\"step\\":91,0,\\\"type\\\":3,0,\\\"amount\\\":921668.95,\\\"oldbalanceOrg\\\":921668.95,\\\"newbalanceOrig\\\":0,0,\\\"oldbalanceDest\\\":0,0,\\\"newbalanceDest\\\":0,0,\\\"isFraud\\\":1.0)"
"(\\"step\\":91,0,\\\"type\\\":0,0,\\\"amount\\\":921668.95,\\\"oldbalanceOrg\\\":921668.95,\\\"newbalanceOrig\\\":0,0,\\\"oldbalanceDest\\\":96395.54,\\\"newbalanceDest\\\":1018064.5,\\\"isFraud\\\":1.0)"
"(\\"step\\":91,0,\\\"type\\\":0,0,\\\"amount\\\":737285.03,\\\"oldbalanceOrg\\\":737285.03,\\\"newbalanceOrig\\\":0,0,\\\"oldbalanceDest\\\":0,0,\\\"newbalanceDest\\\":737285.03,\\\"isFraud\\\":1.0)"
"(\\"step\\":92,0,\\\"type\\\":3,0,\\\"amount\\\":1461930.03,\\\"oldbalanceOrg\\\":1461930.03,\\\"newbalanceOrig\\\":0,0,\\\"oldbalanceDest\\\":0,0,\\\"newbalanceDest\\\":0,0,\\\"isFraud\\\":1.0)"
"(\\"step\\":92,0,\\\"type\\\":0,0,\\\"amount\\\":1461930.03,\\\"oldbalanceOrg\\\":1461930.03,\\\"newbalanceOrig\\\":0,0,\\\"oldbalanceDest\\\":3762496.73,\\\"newbalanceDest\\\":5224426.759999998,\\\"isFraud\\\":1.0)"
"(\\"step\\":93,0,\\\"type\\\":3,0,\\\"amount\\\":886588.02,\\\"oldbalanceOrg\\\":886588.02,\\\"newbalanceOrig\\\":0,0,\\\"oldbalanceDest\\\":0,0,\\\"newbalanceDest\\\":0,0,\\\"isFraud\\\":1.0)"
"(\\"step\\":93,0,\\\"type\\\":0,0,\\\"amount\\\":886588.02,\\\"oldbalanceOrg\\\":886588.02,\\\"newbalanceOrig\\\":0,0,\\\"oldbalanceDest\\\":0,0,\\\"newbalanceDest\\\":886588.02,\\\"isFraud\\\":1.0)"
"(\\"step\\":93,0,\\\"type\\\":3,0,\\\"amount\\\":3206623.3900000001,\\\"oldbalanceOrg\\\":3206623.3900000001,\\\"newbalanceOrig\\\":0,0,\\\"oldbalanceDest\\\":0,0,\\\"newbalanceDest\\\":0,0,\\\"isFraud\\\":1.0)"
"(\\"step\\":93,0,\\\"type\\\":0,0,\\\"amount\\\":3206623.3900000001,\\\"oldbalanceOrg\\\":3206623.3900000001,\\\"newbalanceOrig\\\":0,0,\\\"oldbalanceDest\\\":3206623.3900000001,\\\"newbalanceDest\\\":3449811.899999999,\\\"isFraud\\\":1.0)"
"(\\"step\\":93,0,\\\"type\\\":3,0,\\\"amount\\\":3723738.0600000001,\\\"oldbalanceOrg\\\":3723738.0600000001,\\\"newbalanceOrig\\\":0,0,\\\"oldbalanceDest\\\":0,0,\\\"newbalanceDest\\\":0,0,\\\"isFraud\\\":1.0)"
"(\\"step\\":93,0,\\\"type\\\":0,0,\\\"amount\\\":3723738.0600000001,\\\"oldbalanceOrg\\\":3723738.0600000001,\\\"newbalanceOrig\\\":0,0,\\\"oldbalanceDest\\\":3216391.21,\\\"newbalanceDest\\\":6940129.269999996,\\\"isFraud\\\":1.0)"
"(\\"step\\":94,0,\\\"type\\\":3,0,\\\"amount\\\":2169679.9100000001,\\\"oldbalanceOrg\\\":2169679.9100000001,\\\"newbalanceOrig\\\":0,0,\\\"oldbalanceDest\\\":0,0,\\\"newbalanceDest\\\":0,0,\\\"isFraud\\\":1.0)"
"(\\"step\\":94,0,\\\"type\\\":0,0,\\\"amount\\\":2169679.9100000001,\\\"oldbalanceOrg\\\":2169679.9100000001,\\\"newbalanceOrig\\\":0,0,\\\"oldbalanceDest\\\":0,0,\\\"newbalanceDest\\\":2169679.9100000001,\\\"isFraud\\\":1.0)"
```

Fig 18 : Fraud Data on a new Kafka topic joined with original data after running through ML Model

...ig/zookeeper.propertiesconfig/server.propertiesudData --from-beginning	~/software/kafka — -bash +
rig":45364.25,"nameDest":"\\"M769132147\\","oldbalanceDest":0.0,"newbalanceDest":0.0,"isFraud":0}"	"("step":1,"type":2,"amount":25.12,"nameOrig":"\\"C1257299717\\","oldbalanceOrg":61663.0,"newbalanceOrg":61637.88,"nameDest":"\\"M1474957626\\","oldbalanceDest":0.0,"newbalanceDest":0.0,"isFraud":0)"	"("step":1,"type":2,"amount":17150.89,"nameOrig":"\\"C181252244\\","oldbalanceOrg":61637.88,"newbalanceOrig":144486.99,"nameDest":"\\"M1733022752\\","oldbalanceDest":0.0,"newbalanceDest":0.0,"isFraud":0)"	"("step":1,"type":2,"amount":679502.24,"nameOrig":"\\"C722417467\\","oldbalanceOrg":290.0,"newbalanceOrg":0.0,"nameDest":"\\"C451111351\\","oldbalanceDest":171866.0,"newbalanceDest":3940085.21,"isFraud":0)"

Fig 19: Non-Fraud Data on a new Kafka topic joined with original data after running through ML Model