Machine Learning on streaming data using Kafka

```
!pip install tensorflow-io==0.17.0
!pip install tensorflow==2.4.0
!pip install kafka-python
```

```
!tar -xzf kafka_2.13-2.7.2.tgz
```

```
!./kafka_2.13-2.7.2/bin/zookeeper-server-start.sh -daemon ./kafka_2.13-2.7.2/config/zookeeper
!./kafka_2.13-2.7.2/bin/kafka-server-start.sh -daemon ./kafka_2.13-2.7.2/config/server.proper
!echo "Waiting for 10 secs until kafka and zookeeper services are up and running"
!sleep 10
```

```
Waiting for 10 secs until kafka and zookeeper services are up and running
```

```
!./kafka_2.13-2.7.2/bin/kafka-topics.sh --create --bootstrap-server 127.0.0.1:9092 --replicat
!./kafka_2.13-2.7.2/bin/kafka-topics.sh --create --bootstrap-server 127.0.0.1:9092 --replicat
```

```
Created topic susy-train.
Created topic susy-test.
```

```
!./kafka_2.13-2.7.2/bin/kafka-topics.sh --describe --bootstrap-server 127.0.0.1:9092 --topic
!./kafka_2.13-2.7.2/bin/kafka-topics.sh --describe --bootstrap-server 127.0.0.1:9092 --topic
```

```
Topic: susy-train       PartitionCount: 1       ReplicationFactor: 1    Configs: segment
        Topic: susy-train       Partition: 0    Leader: 0       Replicas: 0     Isr: 0
Topic: susy-test        PartitionCount: 2       ReplicationFactor: 1    Configs: segment
        Topic: susy-test        Partition: 0    Leader: 0       Replicas: 0     Isr: 0
        Topic: susy-test        Partition: 1    Leader: 0       Replicas: 0     Isr: 0
```

```
import os
from datetime import datetime
import time
import threading
import json
from kafka import KafkaProducer
from kafka.errors import KafkaError
from sklearn.model_selection import train_test_split
import pandas as pd
import tensorflow as tf
import tensorflow_io as tfio
```

```
print("tensorflow-io version: {}".format(tfio.__version__))
print("tensorflow version: {}".format(tf.__version__))
```

```
tensorflow-io version: 0.17.0
tensorflow version: 2.4.0
```

```
COLUMNS = [
          #  labels
           'class',
          #  low-level features
           'lepton_1_pT',
           'lepton_1_eta',
           'lepton_1_phi',
           'lepton_2_pT',
           'lepton_2_eta',
           'lepton_2_phi',
           'missing_energy_magnitude',
           'missing_energy_phi',
          #  high-level derived features
           'MET_rel',
           'axial_MET',
           'M_R',
           'M_TR_2',
           'R',
           'MT2',
           'S_R',
           'M_Delta_R',
           'dPhi_r_b',
           'cos(theta_r1)'
           ]
```

```
susy_iterator = pd.read_csv('SUSY.csv.gz', header=None, names=COLUMNS, chunksize=100000)
susy_df = next(susy_iterator)
susy_df.head()
```

	class	lepton_1_pT	lepton_1_eta	lepton_1_phi	lepton_2_pT	lepton_2_eta	lepton_2_
0	0.0	0.972861	0.653855	1.176225	1.157156	-1.739873	-0.874
1	1.0	1.667973	0.064191	-1.225171	0.506102	-0.338939	1.672
2	1.0	0.444840	-0.134298	-0.709972	0.451719	-1.613871	-0.768
3	1.0	0.381256	-0.976145	0.693152	0.448959	0.891753	-0.677
4	1.0	1.309996	-0.690089	-0.676259	1.589283	-0.693326	0.622

```
# Number of datapoints and columns
len(susy_df), len(susy_df.columns)
```

```
(100000, 19)
```

```
# Number of datapoints belonging to each class (0: background noise, 1: signal)
len(susy_df[susy_df["class"]==0]), len(susy_df[susy_df["class"]==1])
```

```
(54025, 45975)
```

```
# Split the dataset

train_df, test_df = train_test_split(susy_df, test_size=0.4, shuffle=True)
print("Number of training samples: ",len(train_df))
print("Number of testing sample: ",len(test_df))

x_train_df = train_df.drop(["class"], axis=1)
y_train_df = train_df["class"]

x_test_df = test_df.drop(["class"], axis=1)
y_test_df = test_df["class"]

# The labels are set as the kafka message keys so as to store data
# in multiple-partitions. Thus, enabling efficient data retrieval
# using the consumer groups.
x_train = list(filter(None, x_train_df.to_csv(index=False).split("\n")[1:]))
y_train = list(filter(None, y_train_df.to_csv(index=False).split("\n")[1:]))

x_test = list(filter(None, x_test_df.to_csv(index=False).split("\n")[1:]))
y_test = list(filter(None, y_test_df.to_csv(index=False).split("\n")[1:]))
```

```
Number of training samples:  60000
Number of testing sample:  40000
```

```
NUM_COLUMNS = len(x_train_df.columns)
len(x_train), len(y_train), len(x_test), len(y_test)
```

```
(60000, 60000, 40000, 40000)
```

```
# Store the train and test data in kafka

def error_callback(exc):
    raise Exception('Error while sendig data to kafka: {0}'.format(str(exc)))

def write_to_kafka(topic_name, items):
  count=0
  producer = KafkaProducer(bootstrap_servers=['127.0.0.1:9092'])
  for message, key in items:
    producer.send(topic_name, key=key.encode('utf-8'), value=message.encode('utf-8')).add_err
```

```
      count+=1
    producer.flush()
    print("Wrote {0} messages into topic: {1}".format(count, topic_name))

write_to_kafka("susy-train", zip(x_train, y_train))
write_to_kafka("susy-test", zip(x_test, y_test))
```

```
    Wrote 60000 messages into topic: susy-train
    Wrote 40000 messages into topic: susy-test
```

```
def decode_kafka_item(item):
  message = tf.io.decode_csv(item.message, [[0.0] for i in range(NUM_COLUMNS)])
  key = tf.strings.to_number(item.key)
  return (message, key)

BATCH_SIZE=64
SHUFFLE_BUFFER_SIZE=64
train_ds = tfio.IODataset.from_kafka('susy-train', partition=0, offset=0)
train_ds = train_ds.shuffle(buffer_size=SHUFFLE_BUFFER_SIZE)
train_ds = train_ds.map(decode_kafka_item)
train_ds = train_ds.batch(BATCH_SIZE)


# Set the parameters

OPTIMIZER="adam"
LOSS=tf.keras.losses.BinaryCrossentropy(from_logits=True)
METRICS=['accuracy']
EPOCHS=10


# design/build the model
model = tf.keras.Sequential([
  tf.keras.layers.Input(shape=(NUM_COLUMNS,)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(256, activation='relu'),
  tf.keras.layers.Dropout(0.4),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.4),
  tf.keras.layers.Dense(1, activation='sigmoid')
])

print(model.summary())
```

```
    Model: "sequential"
    _________________________________________________________________
    Layer (type)                 Output Shape              Param #
    =================================================================
    dense (Dense)                (None, 128)               2432
    _________________________________________________________________
    dropout (Dropout)            (None, 128)               0
    _________________________________________________________________
```

```
dense_1 (Dense)              (None, 256)               33024
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 129
=================================================================
Total params: 68,481
Trainable params: 68,481
Non-trainable params: 0
_________________________________________________________________
None
```

```
# compile the model
model.compile(optimizer=OPTIMIZER, loss=LOSS, metrics=METRICS)
```

```
# fit the model
model.fit(train_ds, epochs=EPOCHS)
```

```
Epoch 1/10
938/938 [==============================] - 35s 36ms/step - loss: 0.5131 - accuracy: 0.74
Epoch 2/10
938/938 [==============================] - 33s 35ms/step - loss: 0.4606 - accuracy: 0.78
Epoch 3/10
938/938 [==============================] - 32s 34ms/step - loss: 0.4550 - accuracy: 0.79
Epoch 4/10
938/938 [==============================] - 33s 35ms/step - loss: 0.4513 - accuracy: 0.79
Epoch 5/10
938/938 [==============================] - 33s 35ms/step - loss: 0.4494 - accuracy: 0.79
Epoch 6/10
938/938 [==============================] - 33s 35ms/step - loss: 0.4485 - accuracy: 0.79
Epoch 7/10
938/938 [==============================] - 33s 34ms/step - loss: 0.4477 - accuracy: 0.79
Epoch 8/10
938/938 [==============================] - 32s 34ms/step - loss: 0.4456 - accuracy: 0.79
Epoch 9/10
938/938 [==============================] - 33s 34ms/step - loss: 0.4449 - accuracy: 0.79
Epoch 10/10
938/938 [==============================] - 34s 36ms/step - loss: 0.4458 - accuracy: 0.79
<tensorflow.python.keras.callbacks.History at 0x7f5f2eba6710>
```

```
test_ds = tfio.experimental.streaming.KafkaGroupIODataset(
    topics=["susy-test"],
    group_id="testcg",
    servers="127.0.0.1:9092",
    stream_timeout=10000,
    configuration=[
        "session.timeout.ms=7000",
```

```
        "max.poll.interval.ms=8000",
        "auto.offset.reset=earliest"
    ],
)

def decode_kafka_test_item(raw_message, raw_key):
  message = tf.io.decode_csv(raw_message, [[0.0] for i in range(NUM_COLUMNS)])
  key = tf.strings.to_number(raw_key)
  return (message, key)

test_ds = test_ds.map(decode_kafka_test_item)
test_ds = test_ds.batch(BATCH_SIZE)

res = model.evaluate(test_ds)
print("test loss, test acc:", res)
```

```
625/625 [==============================] - 14s 22ms/step - loss: 0.4368 - accuracy: 0.79
test loss, test acc: [0.4368078112602234, 0.7974249720573425]
```

```
!./kafka_2.13-2.7.2/bin/kafka-consumer-groups.sh --bootstrap-server 127.0.0.1:9092 --describe
```

```
GROUP           TOPIC           PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG
testcg          susy-test       0          21664           21664           0
testcg          susy-test       1          18336           18336           0
```

```
online_train_ds = tfio.experimental.streaming.KafkaBatchIODataset(
    topics=["susy-train"],
    group_id="cgonline",
    servers="127.0.0.1:9092",
    stream_timeout=10000, # in milliseconds, to block indefinitely, set it to -1.
    configuration=[
        "session.timeout.ms=7000",
        "max.poll.interval.ms=8000",
        "auto.offset.reset=earliest"
    ],
)


def decode_kafka_online_item(raw_message, raw_key):
  message = tf.io.decode_csv(raw_message, [[0.0] for i in range(NUM_COLUMNS)])
  key = tf.strings.to_number(raw_key)
  return (message, key)

for mini_ds in online_train_ds:
  mini_ds = mini_ds.shuffle(buffer_size=32)
  mini_ds = mini_ds.map(decode_kafka_online_item)
  mini_ds = mini_ds.batch(32)
```

```
if len(mini_ds) > 0:
  model.fit(mini_ds, epochs=3)
```

```
Epoch 1/3
32/32 [==============================] - 0s 5ms/step - loss: 0.4638 - accuracy: 0.773
Epoch 2/3
32/32 [==============================] - 0s 5ms/step - loss: 0.4483 - accuracy: 0.777
Epoch 3/3
32/32 [==============================] - 0s 4ms/step - loss: 0.4416 - accuracy: 0.782
Epoch 1/3
32/32 [==============================] - 0s 4ms/step - loss: 0.4510 - accuracy: 0.801
Epoch 2/3
32/32 [==============================] - 0s 4ms/step - loss: 0.4278 - accuracy: 0.810
Epoch 3/3
32/32 [==============================] - 0s 3ms/step - loss: 0.4071 - accuracy: 0.820
Epoch 1/3
32/32 [==============================] - 0s 4ms/step - loss: 0.4618 - accuracy: 0.779
Epoch 2/3
32/32 [==============================] - 0s 4ms/step - loss: 0.4384 - accuracy: 0.790
Epoch 3/3
32/32 [==============================] - 0s 4ms/step - loss: 0.4400 - accuracy: 0.783
Epoch 1/3
32/32 [==============================] - 0s 4ms/step - loss: 0.4514 - accuracy: 0.806
Epoch 2/3
32/32 [==============================] - 0s 4ms/step - loss: 0.4367 - accuracy: 0.801
Epoch 3/3
32/32 [==============================] - 0s 4ms/step - loss: 0.4330 - accuracy: 0.801
Epoch 1/3
32/32 [==============================] - 0s 4ms/step - loss: 0.4517 - accuracy: 0.792
Epoch 2/3
32/32 [==============================] - 0s 5ms/step - loss: 0.4255 - accuracy: 0.806
Epoch 3/3
32/32 [==============================] - 0s 4ms/step - loss: 0.4173 - accuracy: 0.820
Epoch 1/3
32/32 [==============================] - 0s 4ms/step - loss: 0.4359 - accuracy: 0.799
Epoch 2/3
32/32 [==============================] - 0s 4ms/step - loss: 0.4273 - accuracy: 0.807
Epoch 3/3
32/32 [==============================] - 0s 4ms/step - loss: 0.4124 - accuracy: 0.808
Epoch 1/3
32/32 [==============================] - 0s 4ms/step - loss: 0.4461 - accuracy: 0.793
Epoch 2/3
32/32 [==============================] - 0s 4ms/step - loss: 0.4251 - accuracy: 0.797
Epoch 3/3
32/32 [==============================] - 0s 5ms/step - loss: 0.4142 - accuracy: 0.814
Epoch 1/3
32/32 [==============================] - 0s 5ms/step - loss: 0.4728 - accuracy: 0.788
Epoch 2/3
32/32 [==============================] - 0s 5ms/step - loss: 0.4474 - accuracy: 0.802
Epoch 3/3
32/32 [==============================] - 0s 5ms/step - loss: 0.4372 - accuracy: 0.801
Epoch 1/3
32/32 [==============================] - 0s 5ms/step - loss: 0.4776 - accuracy: 0.771
Epoch 2/3
32/32 [==============================] - 0s 5ms/step - loss: 0.4614 - accuracy: 0.778
Epoch 3/3
```

```
32/32 [==============================] - 0s 4ms/step - loss: 0.4413 - accuracy: 0.793
Epoch 1/3
32/32 [==============================] - 0s 6ms/step - loss: 0.4326 - accuracy: 0.802
Epoch 2/3
32/32 [==============================] - 0s 5ms/step - loss: 0.4134 - accuracy: 0.808
```

Task 1: Execute the above code properly with the given dataset.

Task 2: Make a report about,

-> detailed analysis of the code

-> How did you execute the task using Kafka, and why is Kafka important in this machine learning model?

Task 3: Feed a new dataset into Kafka. Utilizing the dataset, train and test your choice of machine learning model and solve any issues that may arise in the code