Train and Deploy Fraud Detection Model by Spark

Yu Qiu

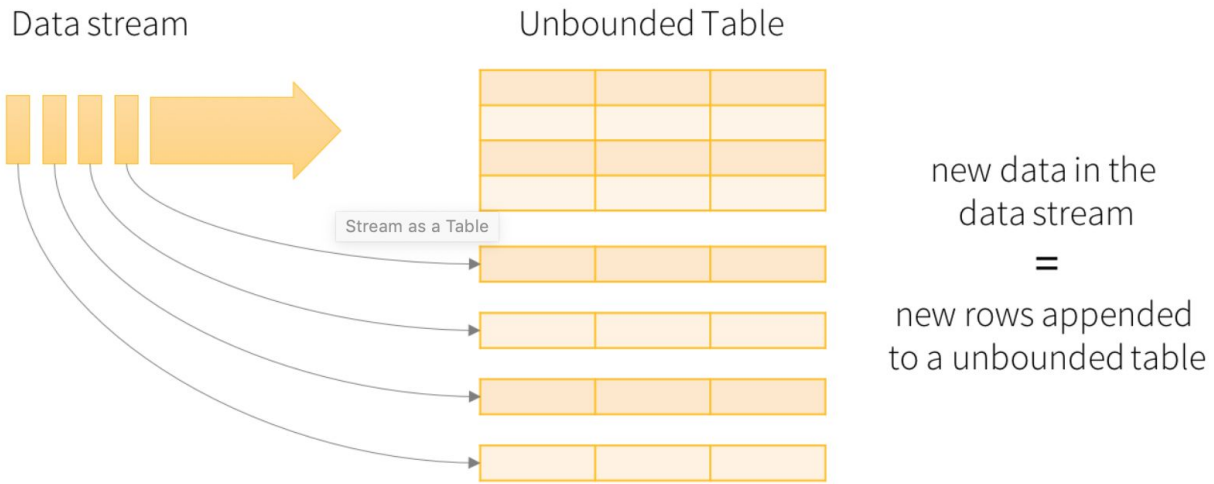
Spark Structured Streaming

Structured Streaming is a scalable and fault-tolerant stream processing engine built on the Spark SQL engine. We can express our streaming computation the same way we expressed a batch computation on static data. The Spark SQL engine will take care of running it incrementally and continuously and updating the final result as streaming data continues to arrive.

Spark Structured Streaming

The key idea in Structured Streaming is to treat a live data stream as a table that is being continuously appended. This leads to a new stream processing model that is very similar to a batch processing model. Internally, by default, Structured Streaming queries are processed using a *micro-batch processing* engine, which processes data streams as a series of small batch jobs.

Spark Structured Streaming



Data stream as an unbounded table

Steps of my work

Step 1: Set up input and output files

Cmd 4

```
1 input_data = "/databricks-datasets/credit-card-fraud/data"
2 output_test_parquet_data = "/tmp/pydata/credit-card-frauld-test-data"
```

Command took 0.02 seconds -- by y.qiu94@hotmail.com at 7/30/2019, 8:36:21 PM on My Cluster

Steps of my work

Step 2: Build ML Pipeline and fit it by training data

```
1  from pyspark.ml import Pipeline
2  from pyspark.sql.functions import col
3
4  pipeline = Pipeline(stages=[oneHot, vectorSizeHint, vectorAssembler, estimator])
5
6  # Split the data into testing and training datasets.
7  # We will shave the test dataset for later
8  train = data.filter(col("time") % 10 < 8)
9  test = data.filter(col("time") % 10 >= 8)
10
11 # Save our data into partitions so we can read them as files
12 (test.repartition(20).write
13   .mode("overwrite")
14   .parquet(output_test_parquet_data))
```

▸ (1) Spark Jobs
▸ train: pyspark.sql.dataframe.DataFrame = [time: integer, amountRange: integer ... 2 more fields]
▸ test: pyspark.sql.dataframe.DataFrame = [time: integer, amountRange: integer ... 2 more fields]

1

Command took 27.34 seconds -- by y.qiu94@hotmail.com at 7/30/2019, 8:36:39 PM on My Cluster

Steps of my Work

Step 3.1: define the streaming data schema

```
from pyspark.sql.types import *
from pyspark.ml.linalg import VectorUDT

schema = (StructType([StructField("time", IntegerType(), True),
                      StructField("amountRange", IntegerType(), True),
                      StructField("label", IntegerType(), True),
                      StructField("pcaVector", VectorUDT(), True)]))
```

Command took 0.03 seconds -- by y.qiu94@hotmail.com at 7/30/2019, 8:36:50 PM on My Cluster

Steps of my work

Step 3.2: read testing data by steam mode and transform it

```
streamingData = (spark.readStream
                 .schema(schema)
                 .option("maxFilesPerTrigger", 1)
                 .parquet(output_test_parquet_data)) # our test data
```

▸ streamingData: pyspark.sql.dataframe.DataFrame = [time: integer, amountRange: integer ... 2 more fields]

Command took 0.36 seconds -- by y.qiu94@hotmail.com at 7/30/2019, 8:36:52 PM on My Cluster

```
from pyspark.sql.functions import *

stream = pipelineModel.transform(streamingData)
```

▸ stream: pyspark.sql.dataframe.DataFrame = [time: integer, amountRange: integer ... 7 more fields]

Command took 0.15 seconds -- by y.qiu94@hotmail.com at 7/30/2019, 8:36:54 PM on My Cluster

Steps of my work

Step 4: Compute several metrics of the model:

```
# define udf

def get_precision_recall(x, y):
  return x / (x + y)

def get_f1_score(x, y):
  return 2 * x * y / (x + y)

get_precision_recall_udf = udf(get_precision_recall, FloatType())
get_f1_score_udf = udf(get_f1_score, FloatType())
```

Command took 0.02 seconds -- by y.qiu94@hotmail.com at 7/30/2019, 8:48:59 PM on My Cluster

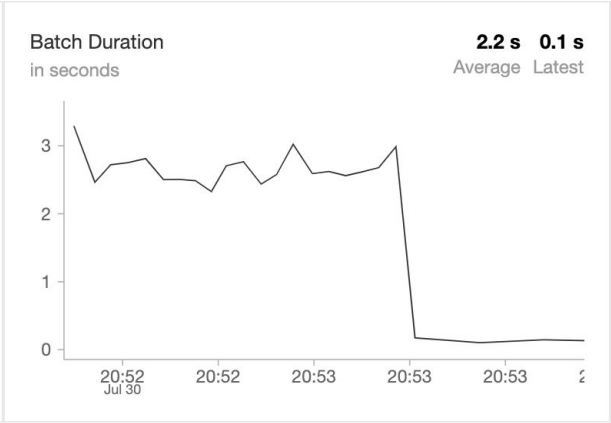
Cmd 29

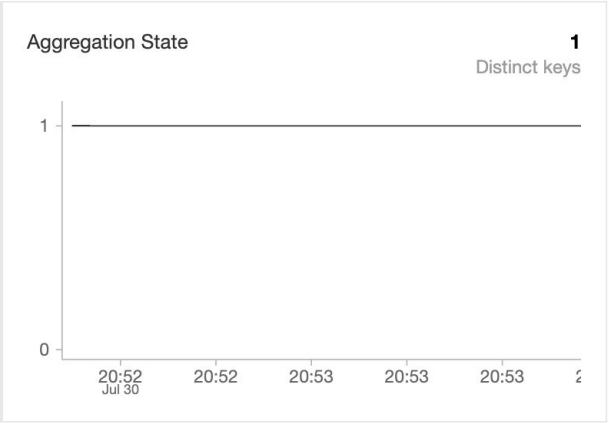
```
streamingMetrics = pipelineModel.transform(streamingData) \
        .select('label', 'prediction') \
        .groupBy() \
        .agg( \
             sum(when((col('prediction') == 1) & (col('label') == 1), 1).otherwise(0)).alias('TP') \
             , sum(when((col('prediction') == 1) & (col('label') == 0), 1).otherwise(0)).alias('FP') \
             , sum(when((col('prediction') == 0) & (col('label') == 0), 1).otherwise(0)).alias('TN') \
             , sum(when((col('prediction') == 0) & (col('label') == 1), 1).otherwise(0)).alias('FN') \
            ) \
        .withColumn('Precision', get_precision_recall_udf(col('TP'), col('FP'))) \
        .withColumn('Recall', get_precision_recall_udf(col('TP'), col('FN'))) \
        .withColumn('F1_Score', get_f1_score_udf(col('Precision'), col('Recall')))
```

▸ streamingMetrics: pyspark.sql.dataframe.DataFrame = [TP: long, FP: long ... 5 more fields]

Result

Dashboard | Raw Data





TP	FP	TN	FN	Precision	Recall	F1_Score
71	9	57131	25	0.8875	0.7395833	0.8068182

Summary

In this demo, I applied spark ML and structured streaming to train a model which can process streaming data. However, for the future works, we can use Spark Structured Streaming to do something more useful. For example, we might aggregate the number of possibly fraudulent transactions for each account over some time period, and give out the security score of each account or transaction.