AWS re:Invent

NOV. 29 – DEC. 3, 2021 | LAS VEGAS, NV

DAT313-R2

Building a machine learning-enabled application with Amazon Aurora

Yahav Biran (he/him)
Principal Solutions Architect
AWS

Steve Dille (he/him)
Senior Product Manager – Technical
AWS

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

- Machine learning and relational databases today
- How Amazon simplifies machine learning
- Use case examples
- Detailed example: Cheat detection in online gaming

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What is machine learning?

- Using technology to discover trends and patterns, and compute mathematical predictive models based on factual past data
- Past data, statistics, and probability theory are key tools used to build machine-learning models and make predictions

Where traditional business analytics aims at answering questions about past events, machine learning aims at answering questions about the possibilities of **future** events

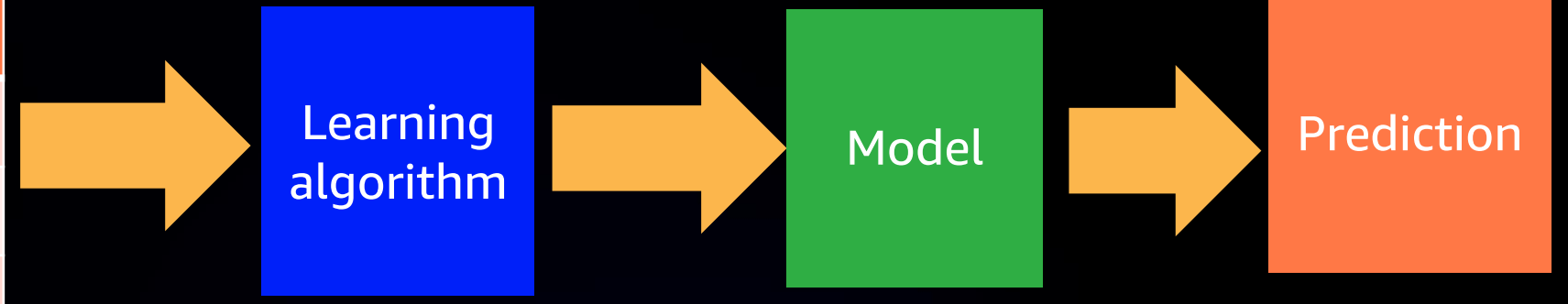
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Supervised machine learning example

Common algorithms:
Linear regression, classification

Age	Gender	Items
35	F	
39	M	Toy

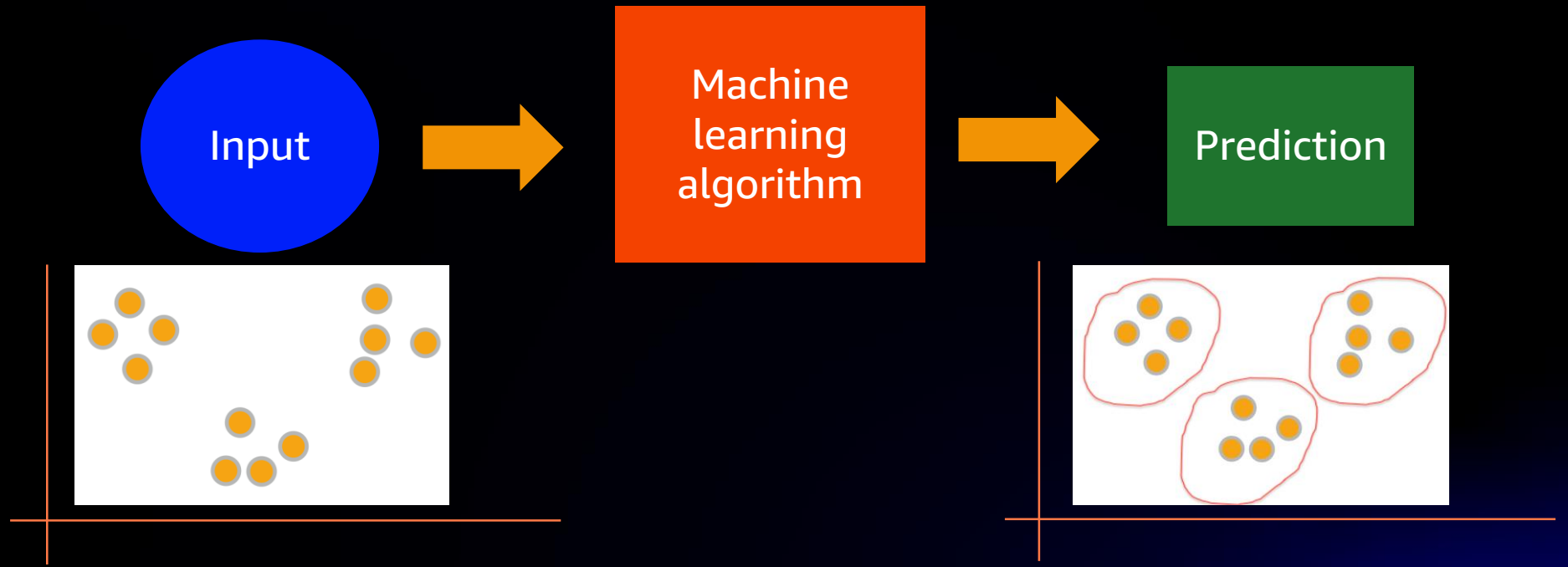
Age	Gender	Purchase Date	Items
30	M	3/20/2017	Toy *
40	M	1/3/2017	Books
….	……	…..	…..



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Unsupervised learning example

- Algorithm (random cut forest) is given mass amounts of data and it must find patterns and relationships between the data; it can then draw inferences from datasets
- Human intervention is not required
  Example: Auto-classification of documents based on context



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Typical machine learning data



Text



Images



Video



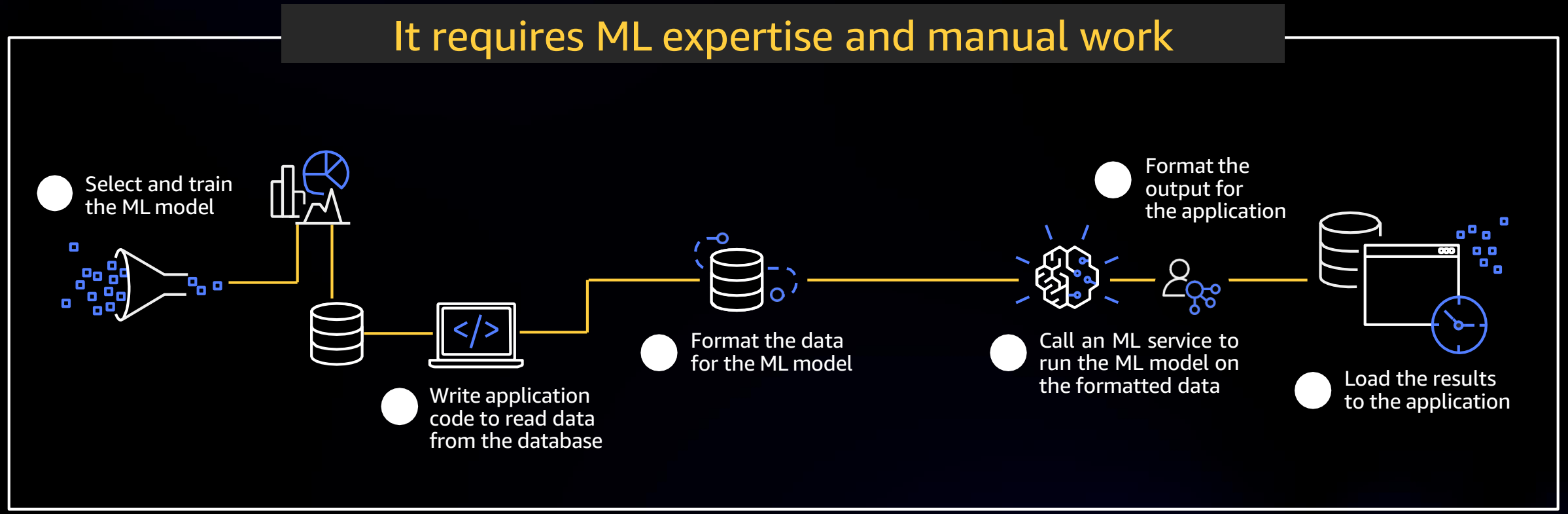
Tabular data

Today’s common scenario: The data is located in a relational database

How can your application use it to make predictions?

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

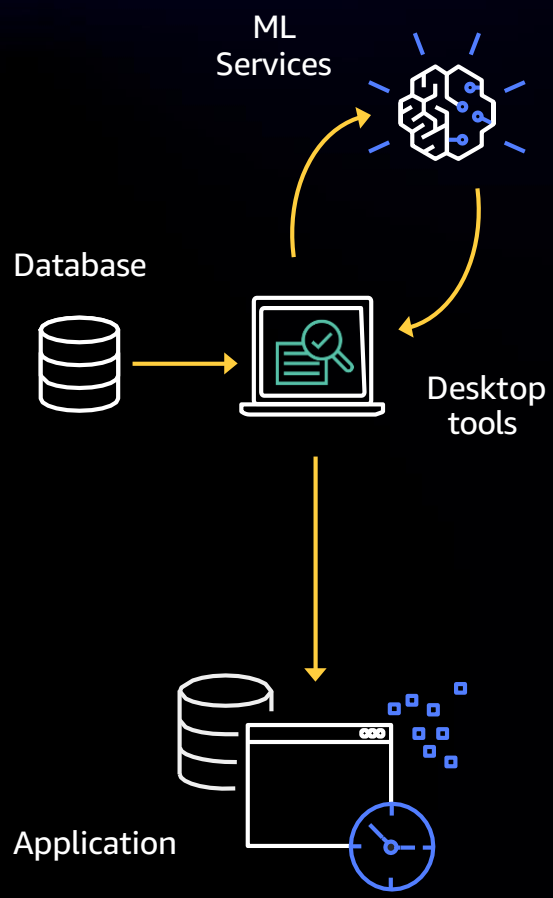
Typical steps for adding ML to an application



Don’t forget the security integration

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

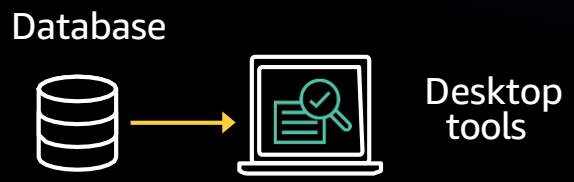
The result: Most ML is done offline



- Data scientist curates and downloads the data manually
- Data preparation, formatting, training, and inference is done using notebooks and custom scripts
- Results in performance problems, high resource usage, and delays
- Less often, there are scheduled or on-demand ML jobs

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Single data source

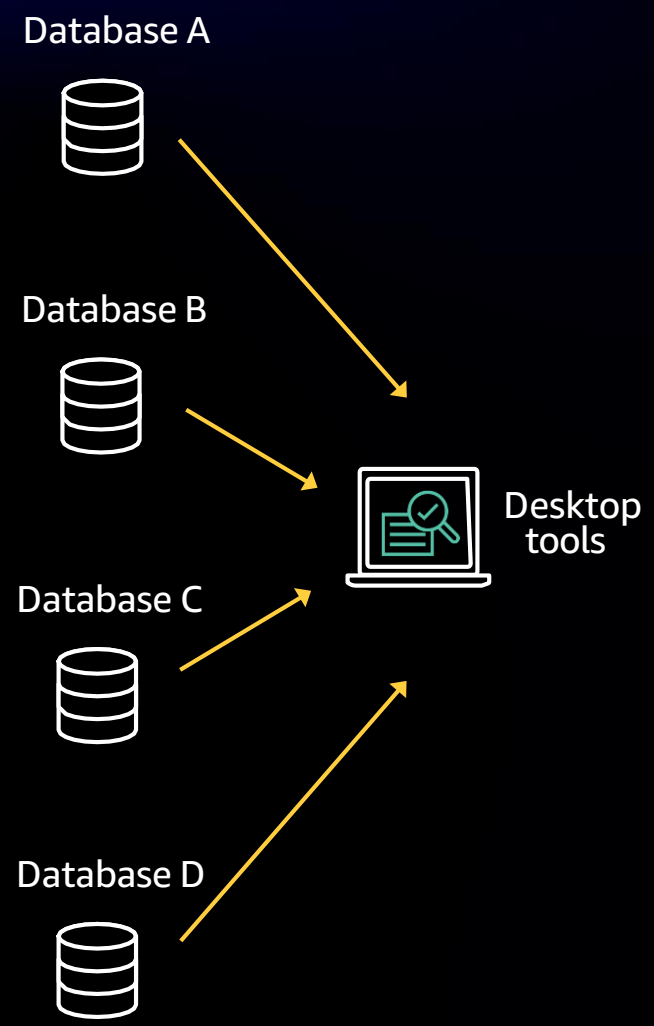


It would be bad enough even if we had a single data source . . .

- Customer service database
- Order management system
- Financial system
- IT management data

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Multiple Amazon Aurora data sources



. . . but it's even worse if there are multiple data sources, like:

- Identity management and authentication
- Transactions
- Customer service

Each source may need a different algorithm, e.g., linear regression for supervised learning or anomaly detection for unsupervised learning

Then we'd want to reconcile them

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Our example is built on Amazon Aurora

Relational database offered by AWS



Speed and availability of high-end commercial databases

Simplicity and cost-effectiveness of open-source databases

Drop-in compatibility with MySQL and PostgreSQL

Simple pay-as-you-go pricing

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Overview of Amazon Aurora machine learning

EASIER, REAL-TIME MACHINE-LEARNING CAPABILITIES FOR RELATIONAL DATA



ML predictions on relational data



Integration with Amazon SageMaker and Amazon Comprehend



Familiar SQL language, no ML expertise



Low-latency, immediate



Security and governance

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

3 ways to implement machine learning

Ranging from no ML experience to bring your own model (BYOM)



Use Amazon Comprehend Sentiment Analysis Function

Use AutoML with Amazon SageMaker AutoPilot

BOYM built with Amazon SageMaker

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Building models with SageMaker Autopilot

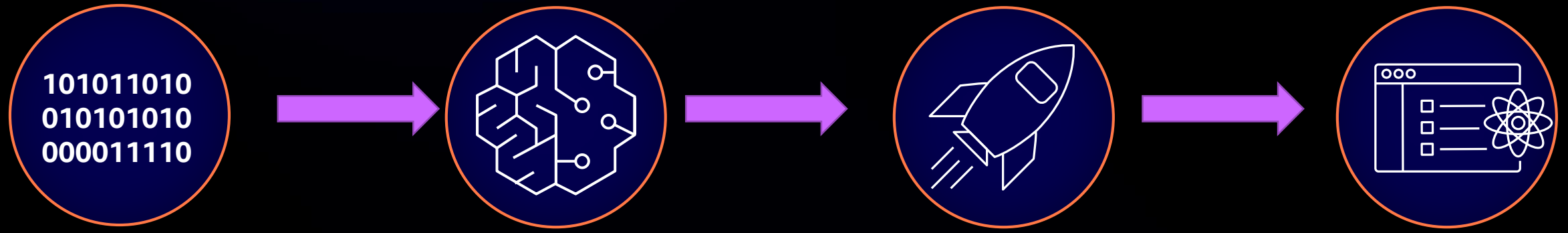


SageMaker Autopilot is natively integrated with SageMaker and is the automated machine-learning capability for tabular data that gives you complete visibility into your ML models

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How Amazon SageMaker Autopilot works



1. Bring data

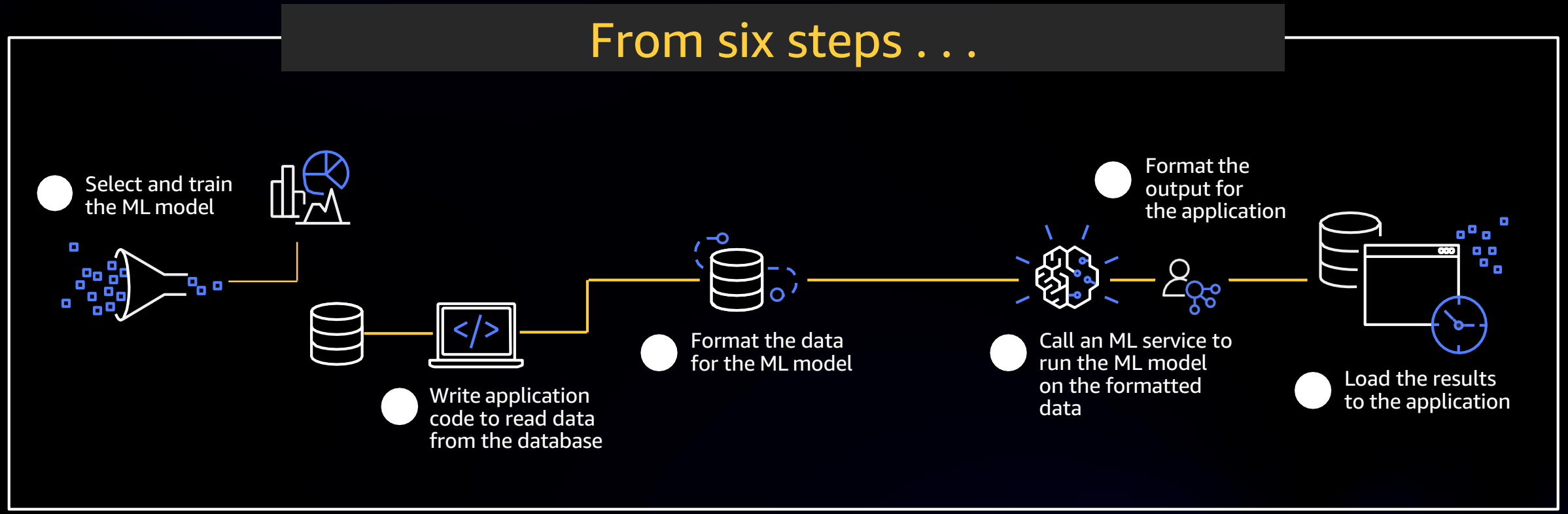
2. Run SageMaker Autopilot

3. Deploy model

4. Make predictions

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

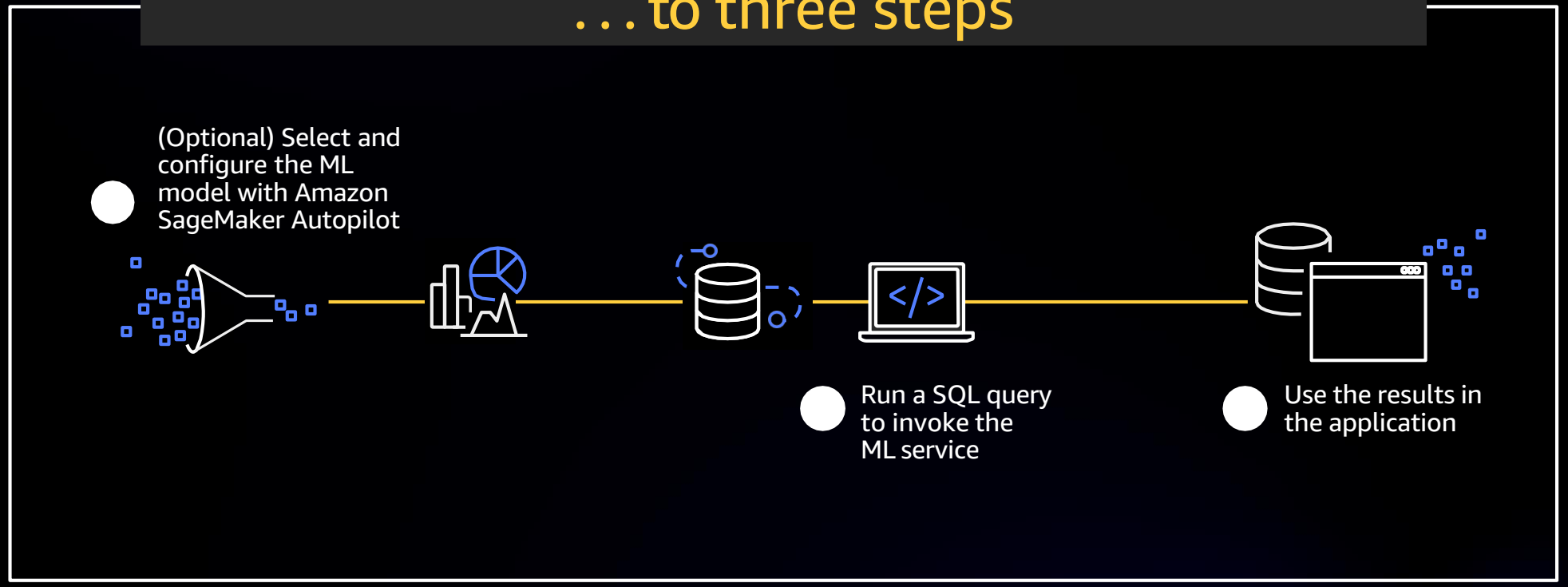
Shortening the process of ML integration



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Shortening the process of ML integration

. . . to three steps



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Calling ML models from Aurora examples

Find suspected fraudulent transactions



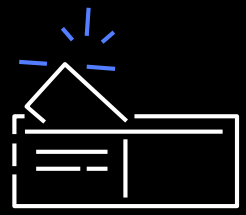
```
CREATE TRIGGER insert_check
BEFORE INSERT ON sales
 FOR EACH ROW BEGIN
  IF is_transaction_fraudulent(column1,
    column2, column3 ...) = 'True' THEN
    rollback; END IF;
 END;
```

Flag negative sentiment comments



```
SELECT * from product_reviews WHERE
aws_comprehend.detect_sentiment
(review_text, 'EN')' = 'NEGATIVE'
```

Sort customers by predicted future spend



```
SELECT * from customers order by
predicted_future_spend (column1,
column2, ...)
```

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Steve Dille
stedille@amazon.com

Yahav Biran
birayaha@amazon.com

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The cheating problem in online gaming



- The game industry generated an estimated $174B worldwide in 2019 (IDC, 2019)
- Player churn grows as cheating grows
  - 60% of online games were negatively impacted by cheaters
  - 77% of players would stop playing a multiplayer game if they thought opponents were cheating (Irdeto, 2019)
- Similar issues may exist in other industries

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Player activity in EmuStar One



We built a massively multiplayer online (MMO) game that enables players to fight, build, explore, and trade goods with each other

- *Emulants* are defined by five personality traits and game events. They can move, forge, dodge etc., and they can transact virtual goods
- https://yahavb.s3-us-west-2.amazonaws.com/EmuStarOne.mp4

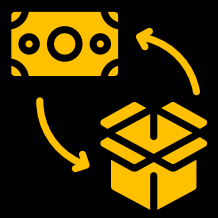
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The EmuStar One is an example of multiple data sources



Player purchase

- Managed and tagged by customer care
- Good candidate for supervised learning

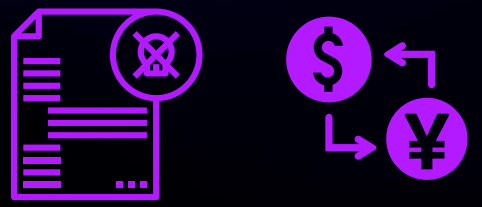


Player trades/events

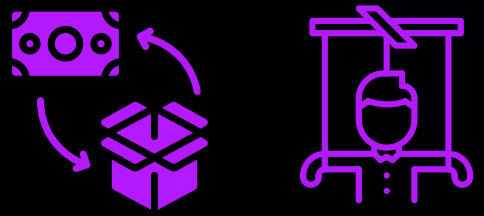
- No easy way to classify, so we can use unsupervised learning

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

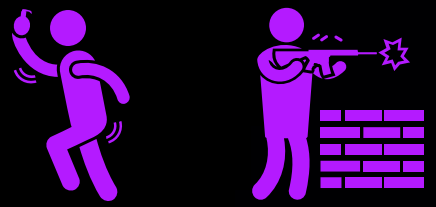
What does cheating look like in the data?



Fraudulent purchase transaction tagged by other players



Players make illegal trades that manipulates the game



Player trade/move anomalies – player walks and shoots in opposite directions while performing other tasks

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How we can stop cheating in EmuStar One?



Forming an anti-cheat team



Rule enforcement – force logout, hard captcha



Cheater will get better

False positives impact non-cheaters

So we need to re-learn

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The ML algorithms we're using on game data



For players transactions:

Supervised linear regression, because most transactions are already classified by customer care and player surveys



For player trades:

Random cut forest (RCF), assuming most player moves are legit so anomalous moves indicate potential cheating

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data preparation

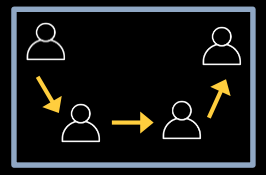
We ran a simulation that generated legit and non legit events:



769,298 authentications



1,358,980 transactions



65,326,500 player moves

Between 0.1% and 3% of each data set was meant to be non-legit

We built the models with:



Jupyter notebooks hosted by SageMaker



S3 to store the training data and model



SageMaker for hosted inference endpoints

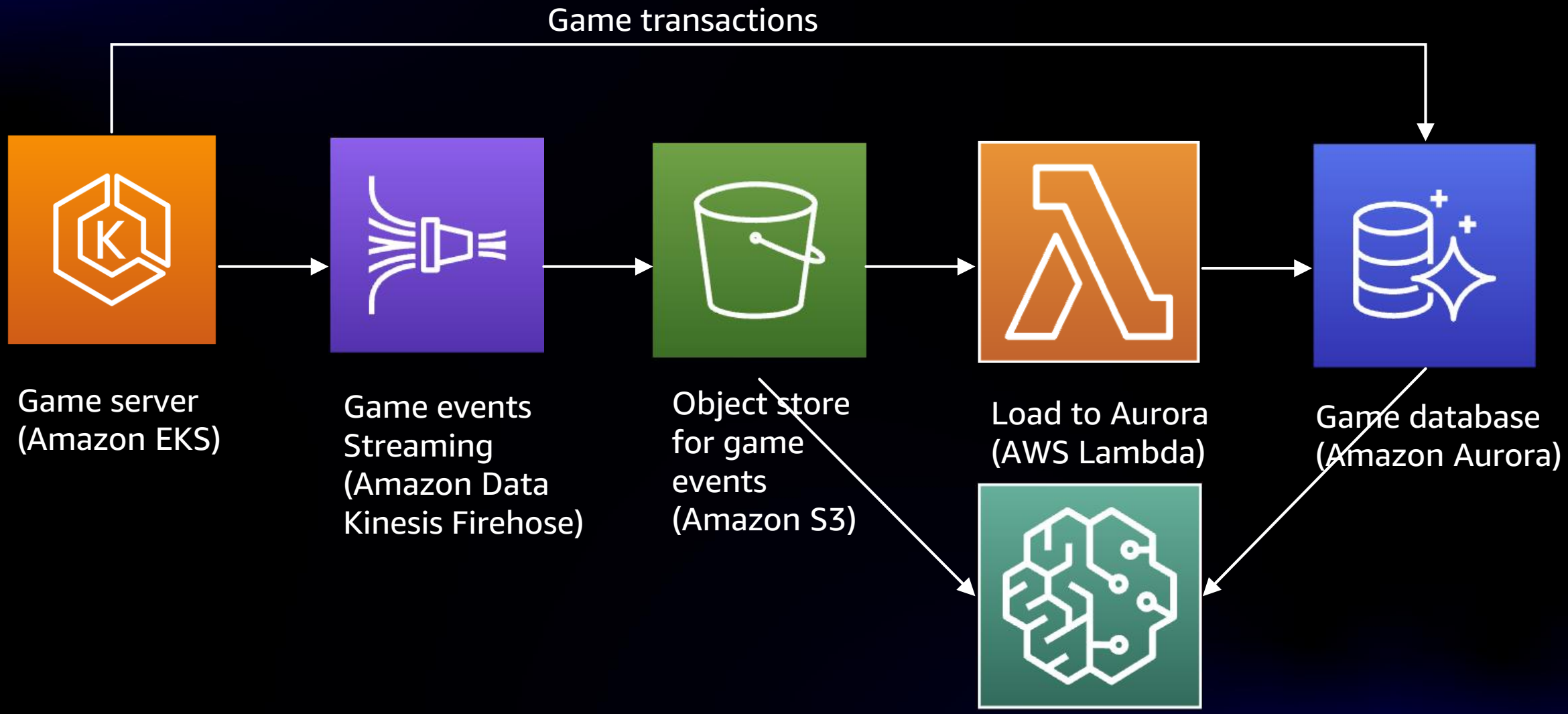


Aurora to invoke the endpoints

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Solution architecture



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

EmuStar One game data: Purchase transactions

```
Player transaction:
{
timestamp:"2020-05-29 19:39:38",
playerGuid:"821f2ac8-5206-44e4-8450-79ab5448cef1",
item:"LootBoxesType2",
user-agent:"Mozilla/5.0 (compatible; MSIE 10.0;
Windows NT 6.1; Trident/5.0; Xbox)"
}
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

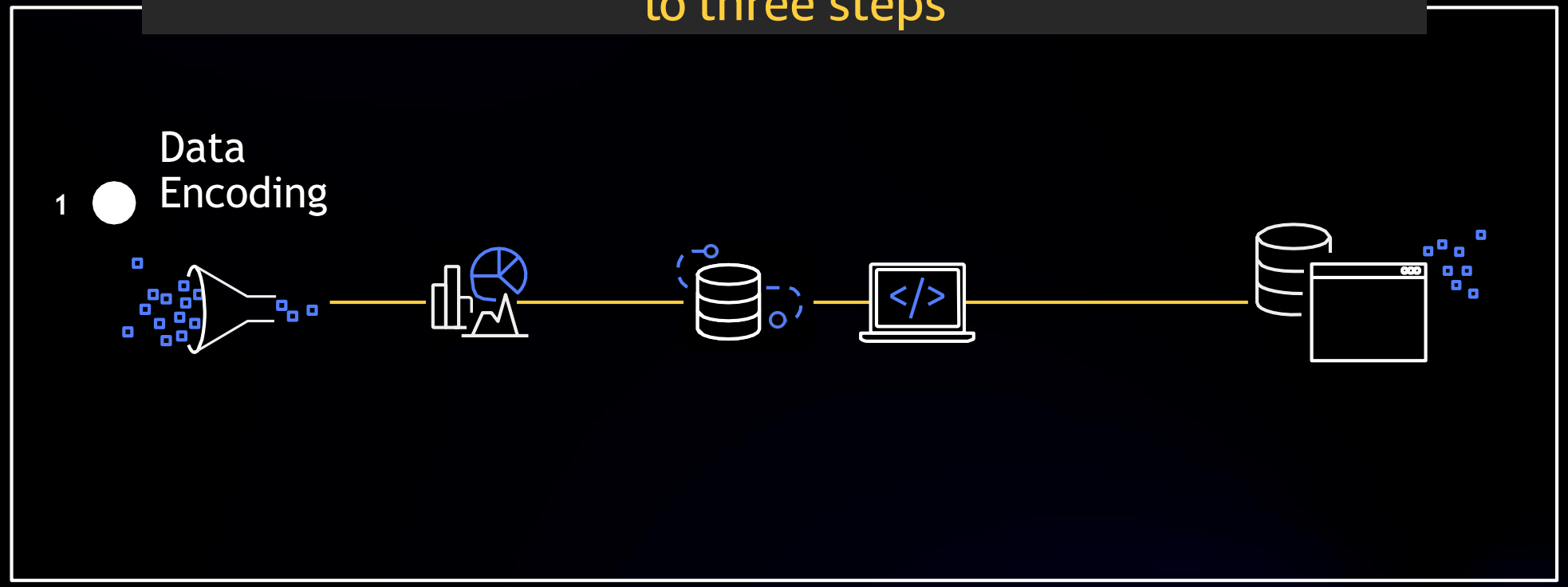
EmuStar One game data: Player trades

```
Player trade:
{
  timestamp:"2020-05-08 09:29:51",
  playerGuid:"463ca444-cdd8-4b64-b6c9-9ddc0de78854",
  playerX:199.0827,
  playerZ:49.08284,
  quadrant:"Quadrant 3",
  sector:"Sector 0,-2",
  event:"TraverseSector",
}
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The ML process: Preparing the data for training

to three steps



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data encoding and transformation

- In general, ML models like numbers – integers, doubles or floats; so we encoded the String attributes
    - We applied the same encoding on the Aurora side

- For example, we transformed:
    - Player move events such as `TraverseSector` or `Travel.Explore`
    - Player move Sector attributes; the game engine generates String values like `Sector 0,-2`
    - Player move quadrant attribute; the game engine generates String values like `Quadrant 3`

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Player moves: Encoding strings to integers

```
import csv
import sys
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
integer_quadrant_encoded = label_encoder.fit_transform(player_data.quadrant)
player_data["quadrant_encoded"]=integer_quadrant_encoded

integer_sector_encoded = label_encoder.fit_transform(player_data.sector)
player_data["sector_encoded"]=integer_sector_encoded

integer_event_encoded = label_encoder.fit_transform(player_data.event)
player_data["event_encoded"]=integer_event_encoded
```

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Player moves: Encoding event names

```
DECLARE enc_cursor
CURSOR FOR
SELECT distinct eventName FROM encounters;
DECLARE CONTINUE HANDLER
...
OPEN enc_cursor;

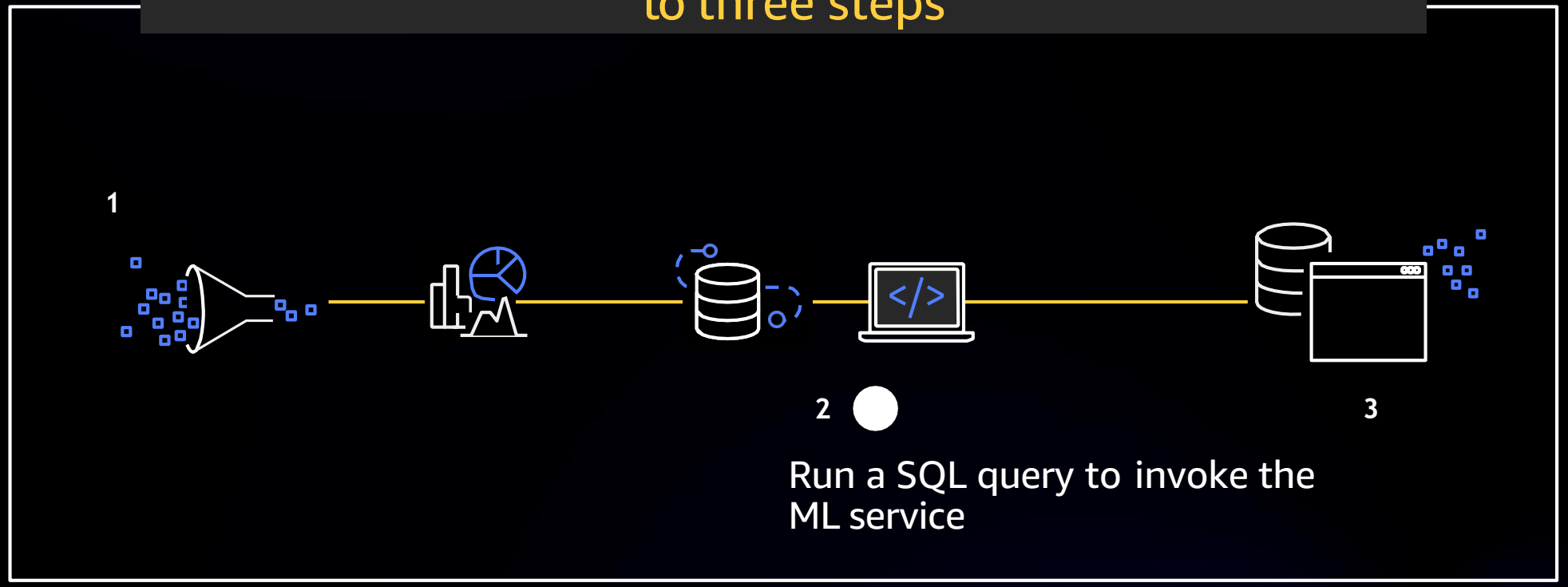
get_encounter: LOOP
FETCH enc_cursor INTO c_eventName;
IF finished =1 THEN
LEAVE getEncounter;
END IF;
select encoded_event+1 into encoded_event;
update encounters set encoded_event=encoded_event where
eventName=c_eventName;
END LOOP get_encounter;
CLOSE enc_cursor;
```

Instead of using libraries like sklearn, we use this to encode event names

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The ML process: Querying the model

to three steps



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Query the model

- We’ll create materialized views for purchase transactions and trades
- Based on the characteristics of cheating in our game, we find cheaters via the following steps:
    - Look for suspicious transactions
    - Look for suspicious trades
    - Then see the nature of the trades for more context

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Materialized view for transactions

```
CREATE table trs_cheat_mv AS
SELECT t.timestamp,
        t.playerGuid
FROM
    (SELECT playerGuid,
        timestamp,
        trans_cheat_score(month,
        day,
        hour,
        minute,
        name_encoded,
        uagent) cls
    FROM transactions t) AS t
WHERE cls>0;
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Materialized view for player moves

```
CREATE table move_cheat_mv AS
SELECT t.timestamp,
        t.playerGuid
FROM
    (SELECT timestamp,
        playerGuid,
        move_cheat_score(playerX,
        playerZ,
        quadrant,
        sector,
        event) cls
    FROM dist_player_moves) AS t
WHERE cls>2;
```

← We figured that an anomaly score greater than 2 indicates a suspicious move

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Look for suspicious transactions

```
mysql> select distinct t.timestamp,t.playerGuid from (select
playerGuid,timestamp,trans_cheat_score(month,day,hour,minute,name_encoded,uagent)
cls from transactions t) as t where cls>0 and DATE(timestamp) between '2021-05-15'
and '2021-05-18';
+---------------------+--------------------------------------+
| timestamp           | playerGuid                           |
+---------------------+--------------------------------------+
| 2020-05-16 15:38:32 | 46fb2825-dd04-443e-9357-d6418b390c26 |
| 2020-05-15 16:59:13 | e474dfa0-f269-41cc-83e4-43315110f375 |
| 2020-05-16 03:32:56 | 20a74aa3-eff2-4d9a-b9c7-a122d661c5c3 |
| 2020-05-15 03:09:02 | 3cbbfebe-e474-4866-b0d6-d62a93fb7a1e |
| 2020-05-16 06:15:14 | 3cbbfebe-e474-4866-b0d6-d62a93fb7a1e |
| 2020-05-18 04:45:01 | 5c3d087a-e823-48c7-8a9b-4eea94772659 |
| 2020-05-18 00:26:49 | 52e5ec16-d5e2-4dee-a078-1464f52c7c48 |
| 2020-05-15 03:23:29 | d5959f4e-4576-4e82-911d-a7656691b3f9 |
| 2020-05-15 07:02:47 | cdda0082-2934-4dca-b00f-0fbc32a2025e |
| 2020-05-16 23:22:45 | cdda0082-2934-4dca-b00f-0fbc32a2025e |
| 2020-05-18 21:52:19 | 6c497f82-dc2c-4c8f-aeb8-b2d1a49199aa |
| 2020-05-18 07:48:23 | a3186242-9cdb-4686-8b3a-c7a22eb192de |
| 2020-05-17 11:15:26 | 12d84e0a-15a3-49f3-a90e-d329a7e30661 |
+---------------------+--------------------------------------+
  13 rows in set (0.12 sec)
```

← Remember this player

← and this player

aws

Let’s see if these players made suspicious trades/moves

```
mysql> select distinct timestamp,playerGuid from (select
timestamp,playerGuid,playerX,playerZ,quadrant,sector,event,move_cheat_score(playerX,
playerZ,quadrant,sector,event) cls from dist_player_moves where DATE(timestamp) between
'2021-05-15' and '2021-05-18') as t where cls>2;
+---------------------+--------------------------------------+
| timestamp           | playerGuid                           |
+---------------------+--------------------------------------+
| 2020-05-15 07:08:25 | cdda0082-2934-4dca-b00f-0fbc32a2025e |
| 2020-05-16 02:20:30 | 7695a00b-ef68-4890-befd-e535eab7c925 |
| 2020-05-17 11:02:40 | 12d84e0a-15a3-49f3-a90e-d329a7e30661 |
| 2020-05-17 15:32:47 | 3d143db5-6d5d-4d4c-b328-97a9fe49cb76 |
| 2020-05-18 00:41:46 | aee05e11-676a-4b00-82df-0fea59fbaed9 |
| 2020-05-18 14:15:32 | 2b3292d1-aa5a-4912-981c-aebdf905cc9f |
+---------------------+--------------------------------------+
 6 rows in set (3.28 sec)
```

Both of our “friends” show up in the suspicious moves list.

One of them has the dubious distinction of cheating in authentication AND transactions AND player moves!

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Is there anything special about these moves?

```
mysql> select distinct timestamp,playerGuid,quadrant,sector from (select
timestamp,playerGuid,playerX,playerZ,quadrant,sector,event,move_cheat_score(playerX,play
erZ,quadrant,sector,event) cls from dist_player_moves where DATE(timestamp) between
'2021-05-15' and '2021-05-18') as t where cls>2;
+---------------------+--------------------------------------+----------+--------+
| timestamp           | playerGuid                           | quadrant | sector |
+---------------------+--------------------------------------+----------+--------+
| 2020-05-15 07:08:25 | cdda0082-2934-4dca-b00f-0fbc32a2025e | 3        | 35     |
| 2020-05-16 02:20:30 | 7695a00b-ef68-4890-befd-e535eab7c925 | 3        | 35     |
| 2020-05-17 11:02:40 | 12d84e0a-15a3-49f3-a90e-d329a7e30661 | 3        | 35     |
| 2020-05-17 15:32:47 | 3d143db5-6d5d-4d4c-b328-97a9fe49cb76 | 3        | 35     |
| 2020-05-18 00:41:46 | aee05e11-676a-4b00-82df-0fea59fbaed9 | 3        | 35     |
| 2020-05-18 14:15:32 | 2b3292d1-aa5a-4912-981c-aebdf905cc9f | 3        | 35     |
+---------------------+--------------------------------------+----------+--------+
6 rows in set (2.82 sec)
```

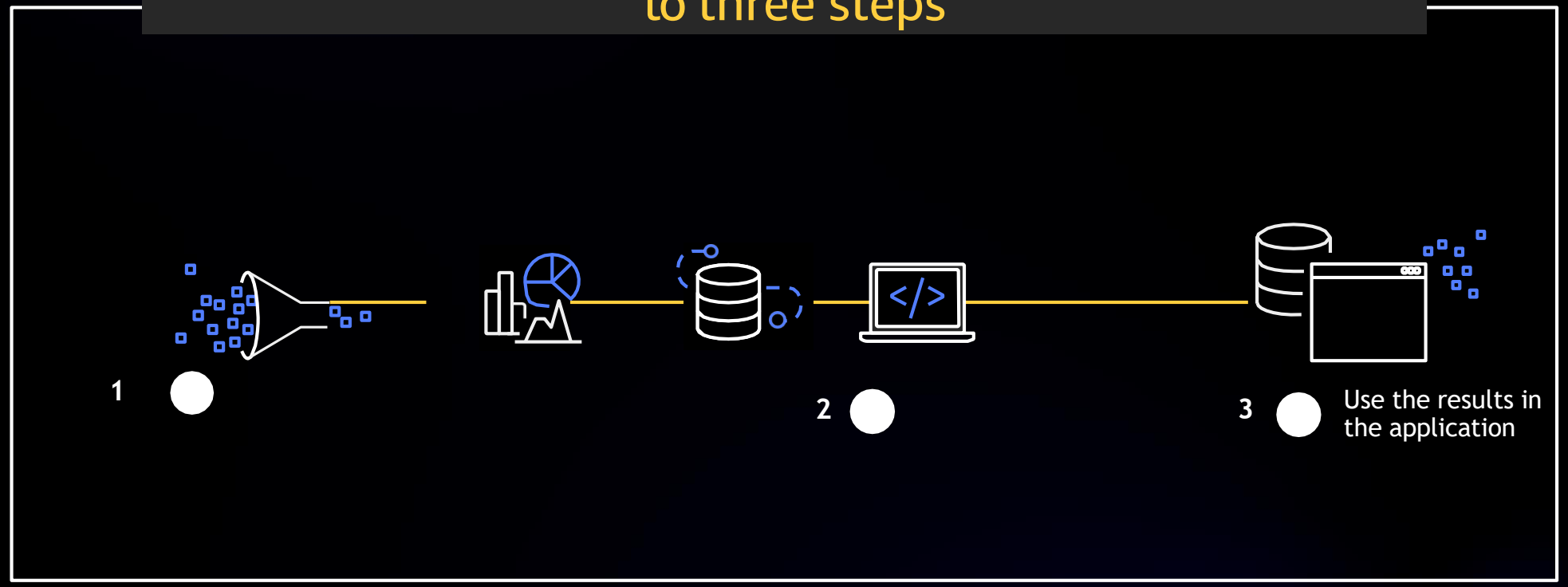
We see that something happened in quadrant 3, sector 35.

Our investigation will continue by simulating or playing in this game location.

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The ML process: Optimizing the application performance

to three steps



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How was our query processing optimized?

Let's see how our calls to SageMaker were optimized. First, reset counters.

```
mysql> flush status;
Query OK, 0 rows affected (0.00 sec)

mysql> show global status like '%aurora_ml%';
+-----------------------------+-------+
| Variable_name | Value |
+-----------------------------+-------+
| Aurora_ml_actual_request_cnt | 0 |
| Aurora_ml_actual_response_cnt | 0 |
| Aurora_ml_cache_hit_cnt | 0 |
| Aurora_ml_logical_request_cnt | 0 |
| Aurora_ml_logical_response_cnt | 0 |
| Aurora_ml_retry_request_cnt | 0 |
| Aurora_ml_single_request_cnt | 0 |
+-----------------------------+-------+
7 rows in set (0.00 sec)
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How was our query processing optimized?

```
mysql> select t.playerGuid from (select playerGuid,
trans_cheat_score(month,day,hour,minute,name_encoded,uagent) cls from transactions t) as
t where cls>0;
....
| 1f310857-8fdd-4870-9c7d-ef62cc2bdd70 |
| e123d51c-edd4-442b-a420-731a593cc75a |
+--------------------------------------+
1750 rows in set (8.90 sec)
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

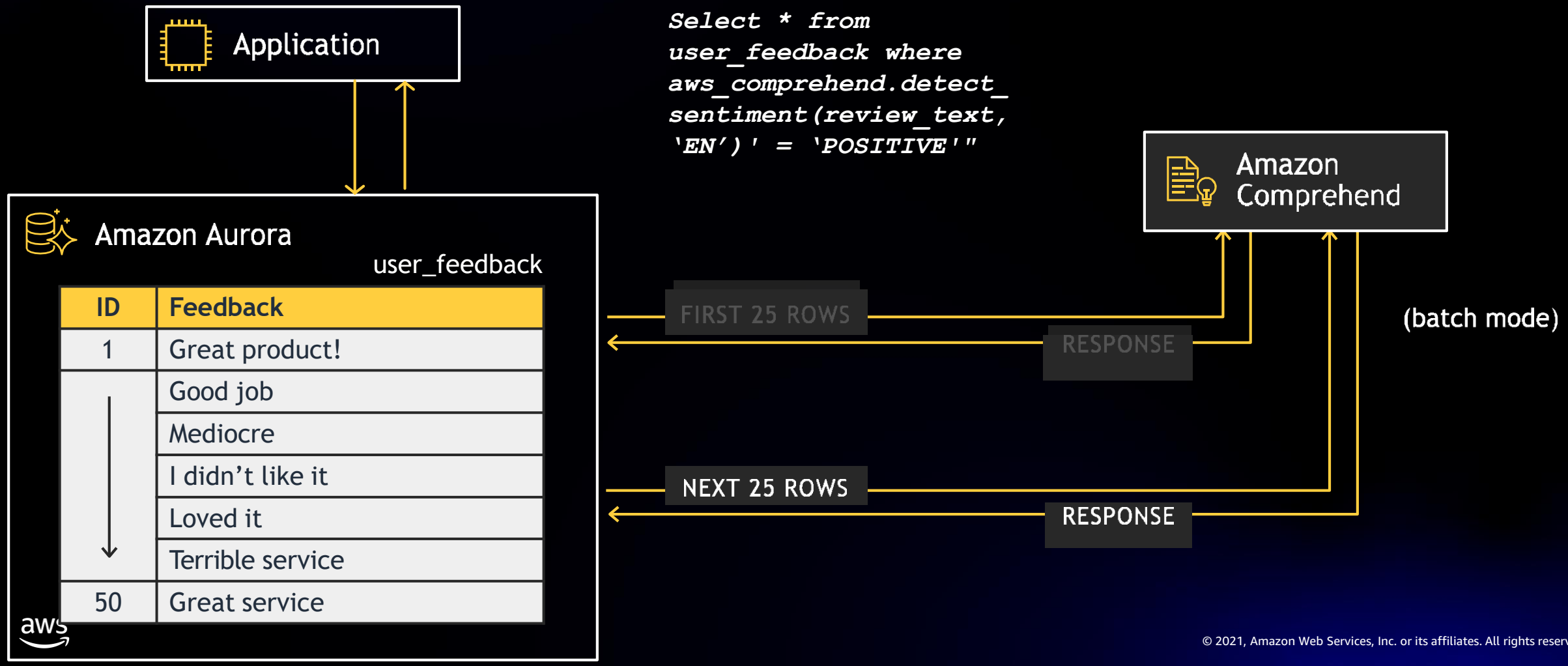
How was our query processing optimized?

```
mysql> show global status like '%aurora_ml%';
+--------------------------+-------+
| Variable_name | Value |
+--------------------------+-------+
 |  Aurora_ml_actual_request_cnt              | 1351    |
 |  Aurora_ml_actual_response_cnt | 1351 |
 |  Aurora_ml_cache_hit_cnt | 1229584 |
 |  Aurora_ml_logical_request_cnt | 1358980                    |
 |  Aurora_ml_logical_response_cnt | 129396                    |
 |  Aurora_ml_retry_request_cnt | 0 |
 |  Aurora_ml_single_request_cnt | 0 |_____
 +-------------------------------------+                      +
 7  rows in set (0.00 sec)
```

Transactions table has 1.3M records, and 1.2M of them hit the cache (no SageMaker calls for these rows). 1351 actual requests were made.

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Under the hood: optimized ML query processing in Aurora



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

For more information

Our examples on GitHub: https://github.com/aws-samples/amazon-aurora-call-to-amazon-sagemaker-sample

Amazon Aurora: https://aws.amazon.com/rds/aurora

Aurora machine learning: https://aws.amazon.com/rds/aurora/machine-learning

Amazon SageMaker: https://aws.amazon.com/sagemaker

Amazon AutoPilot Video: https://www.youtube.com/watch?v=KrGIWHoOfKU

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Steve Dille
stedille@amazon.com

Yahav Biran
birayaha@amazon.com

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.