DATA+AI SUMMIT 2022

Multimodal Deep Learning at Scale - Learning from Catalogs at Mirakl

Arthur Delaitre
Data Scientist, Mirakl

Sang-Hoon Yoon
Data Scientist, Mirakl

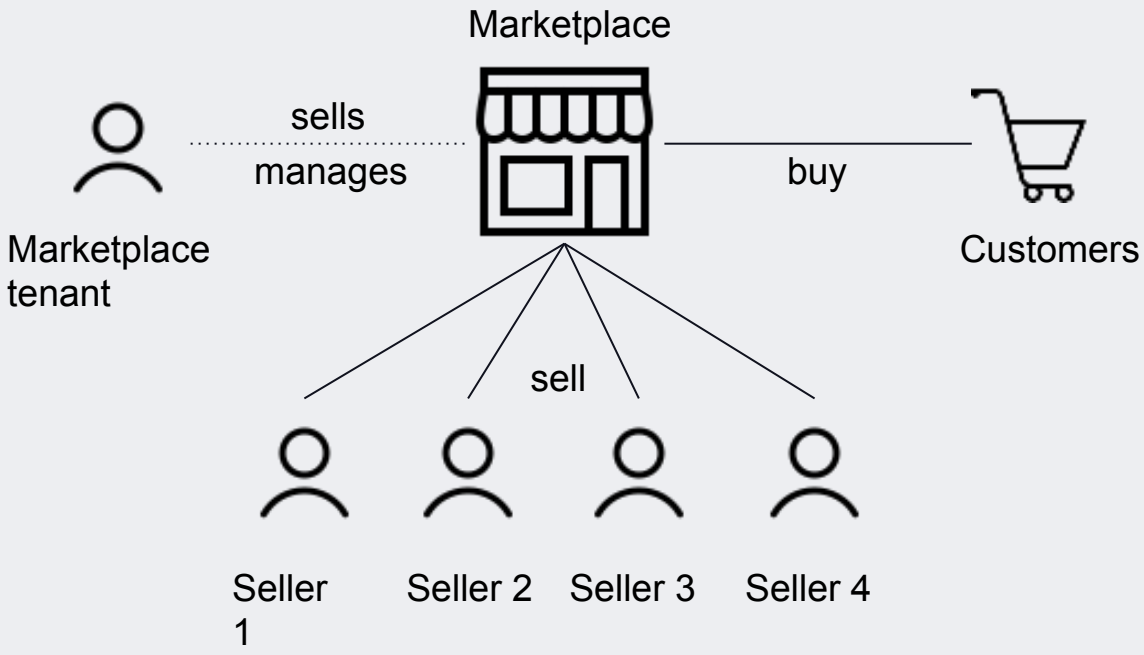
Could not come to present his work :

Milton Minervino
Data Scientist, Mirakl

ORGANIZED BY databricks

What is a marketplace ?

Marketplace: a platform where multiple providers sell products and services



Some Mirakl powered marketplaces :

Kroger

Carrefour

AIRBUS HELICOPTERS

macy's

BED BATH & BEYOND

Who are we ?

- 🚀 Mirakl is the leader marketplace SaaS platform that empowers both B2B and B2C organizations to launch and grow an enterprise marketplace at scale.

- 👩‍🚀 Our Data Science team aims to develop and industrialize AI features to improve user experience (automatic categorization and mapping, catalog cleaning, customer care support, …)

Main topics

1. Catalog data
   A large and diverse source of data ideal for multimodal learning
2. Catalog use cases
   Key use cases: Categorization and Duplicates finding
3. Categorize with multimodal product embeddings
   Multimodal product embeddings: product2vec
4. Curate the catalog by removing duplicates
   A reliable and coherent product catalog
5. Key takeaways

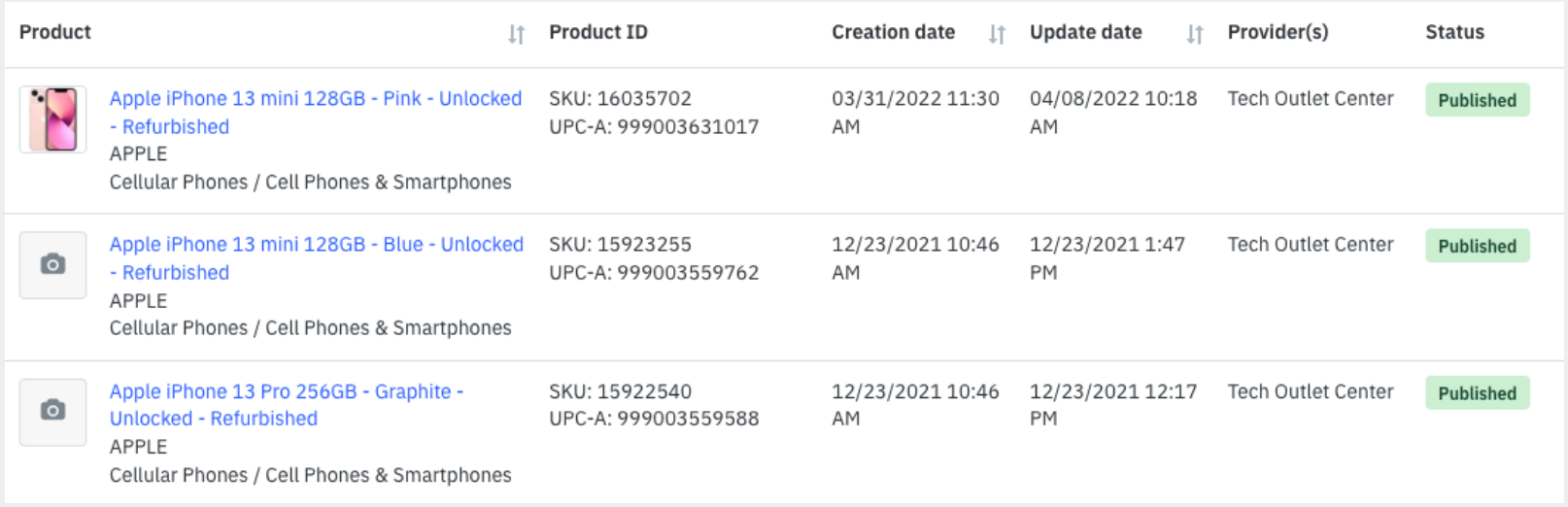
Catalog data

A large and diverse source of data ideal for multimodal learning

What is a catalog in a marketplace ?

In Mirakl, the catalog is **a database containing all products data of the marketplace**

Multiple sellers can upload their products : **Mirakl Catalog Manager** ensures that **everything is represented as a single, coherent and reliable catalog**.



Figures: Mirakl Catalog Management

Key numbers

Number of products*

310 M

Number of categories

140 k

Number of marketplaces

+300

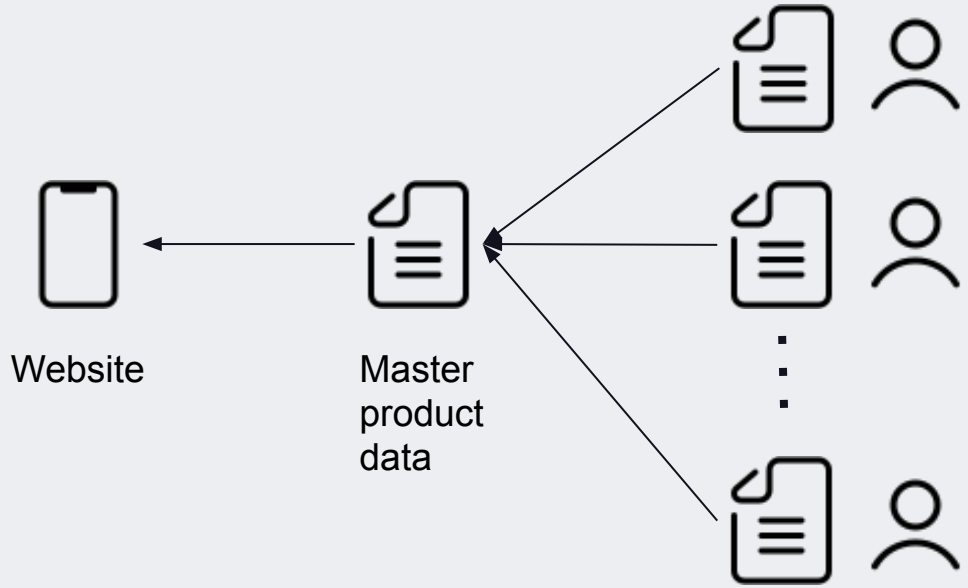
Number of sellers

+200 k

The kind of data we have

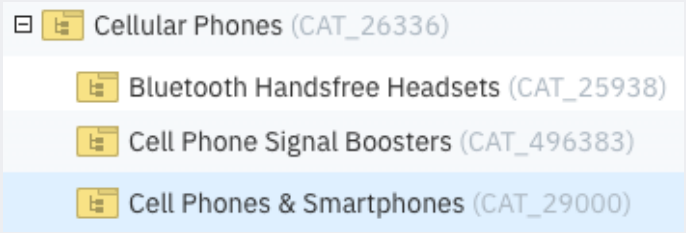
Product data contain:

- **images** (product pictures)
- **texts**
- tabular data (category, size, storage capacity, …)



Different product data from sellers

The kind of data we have



Title: iPhone 13 6,1” 5G 128 Go Double SIM

Description: This is an iPhone …

Color: Midnight

Dimensions: 146,7 x 71,5 x 7,65 mm

Screen: 6.1” OLED

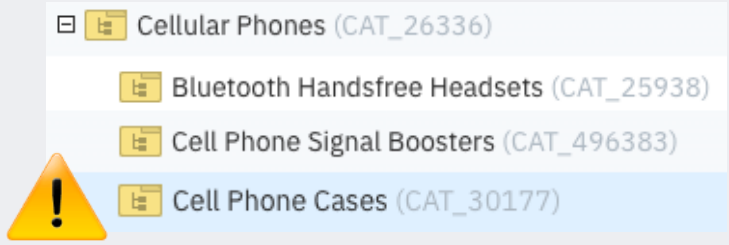
Processor: A15 Bionic Chip

Camera resolution: 12MP (Ultra Wide)

Weight: 173g

Storage: 128GB

The kind of data we have



Title: iPhone 13 6,1” 5G 128 Go Double SIM

Description: Le dernier smartphone d’Apple …

Color: Gold

Dimensions: 146,7 x 71,5 x 7,65 mm

Screen: NaN

Processor: A15 Bionic Chip

Camera resolution: 12MP (Ultra Wide)

Weight: 6,10 ounces

Storage: 128-512 GB

Recurring issues with product data

The recurring issues:

- **Data quality**
  - Product with **missing/ corrupted data**
  - Sellers may **mislabel**

- **Multiple standards**
  - Different **metric systems**
  - Different **languages**
  - Heterogeneous **category trees**

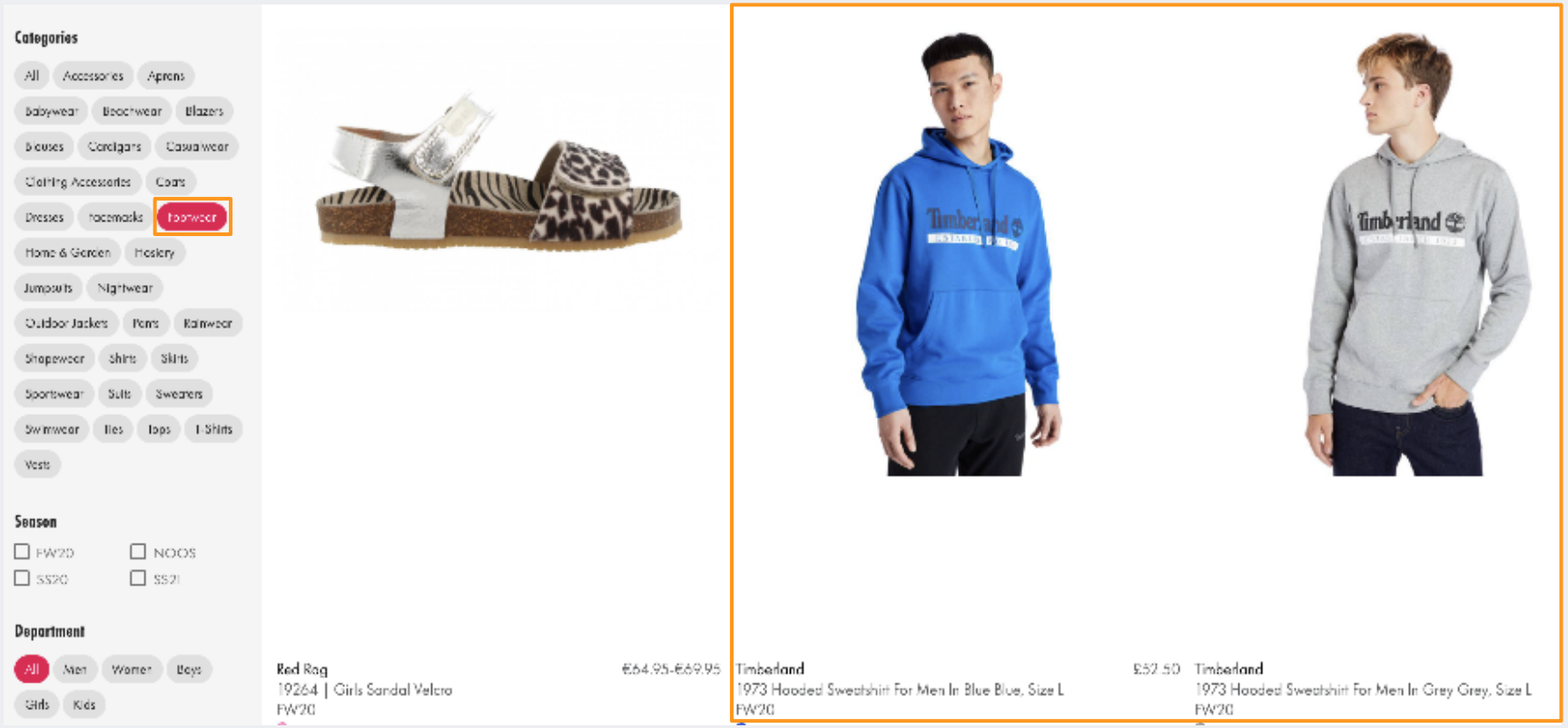
These product data issues prevent marketplaces from proposing a reliable catalog

Catalog use cases

Categorization and Duplicates finding

Use case 1 : **Product Categorization**

Some products on the marketplace are assigned to a wrong category



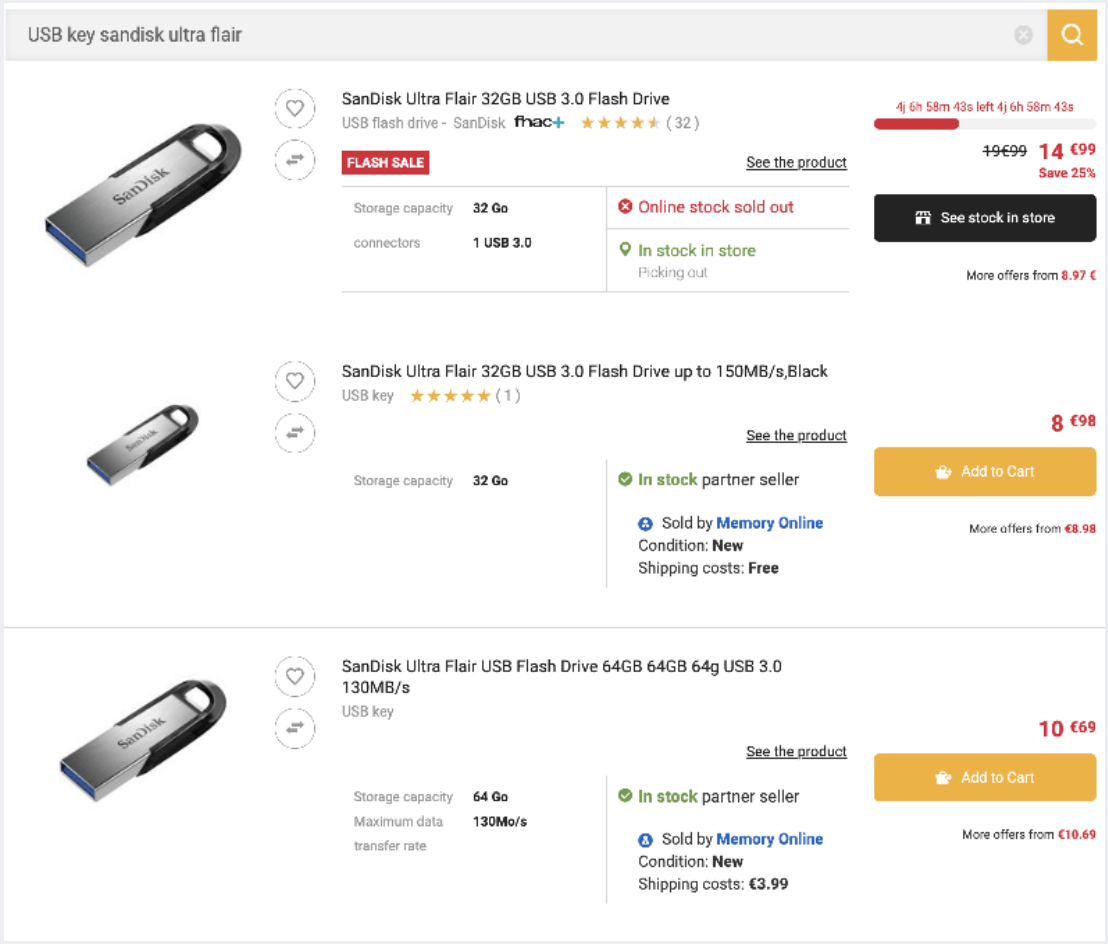
Sweaters in the wrong category “footwear”

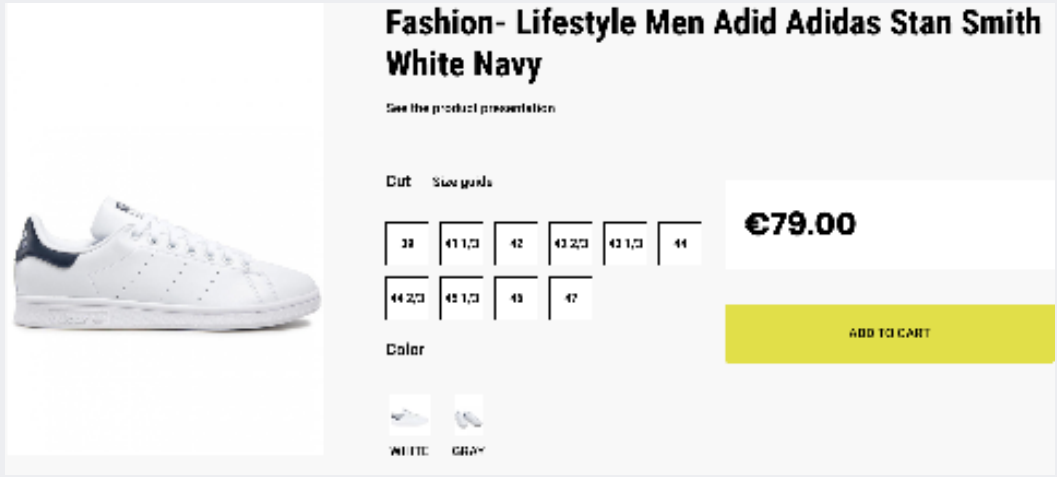
Consequences:

- Poor UX
- Hidden products

Goal: predict the category with product data such as image, texts, color, size, etc

Use case 2 : **Product duplicates finding**





Goal: find the product data referring to the same product to clean the catalog

Categorization

Categorize with multimodal product embeddings

Categorization model architecture

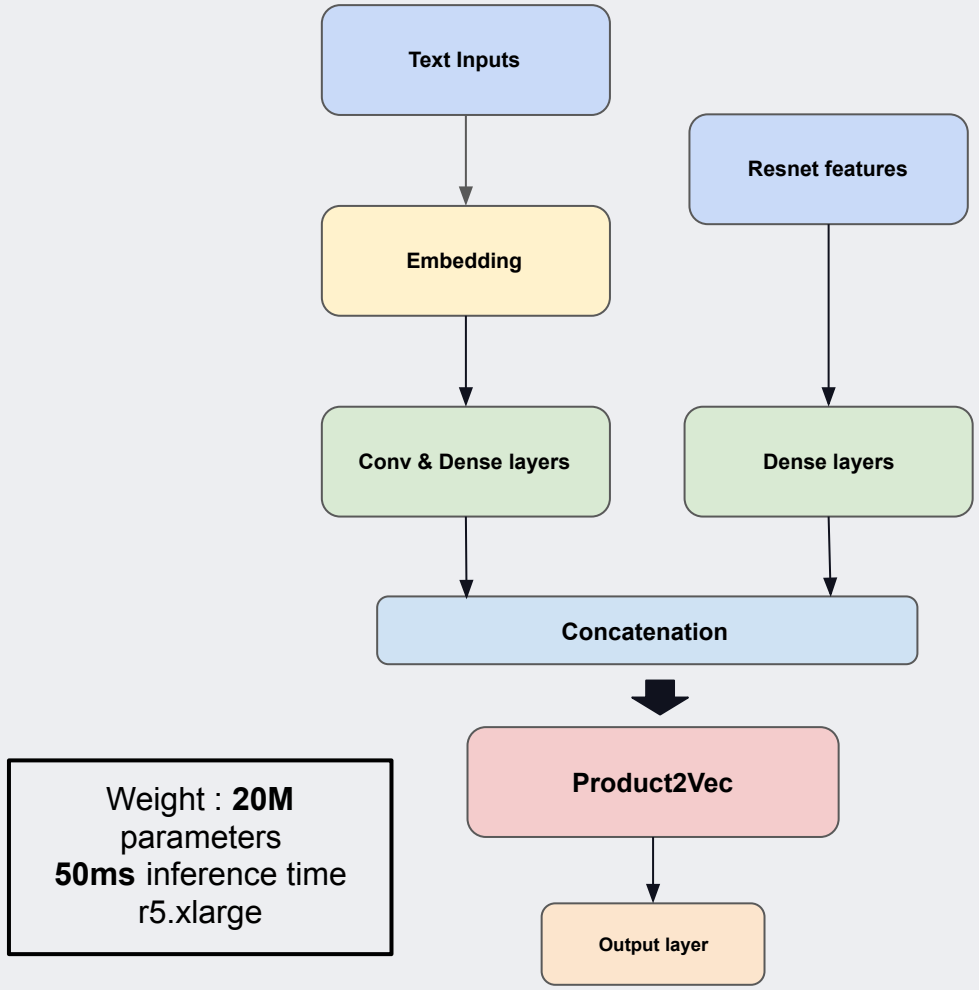


Figure: Categorization model architecture

Multimodal categorization model that corrects mislabeled product categories

Requirements:

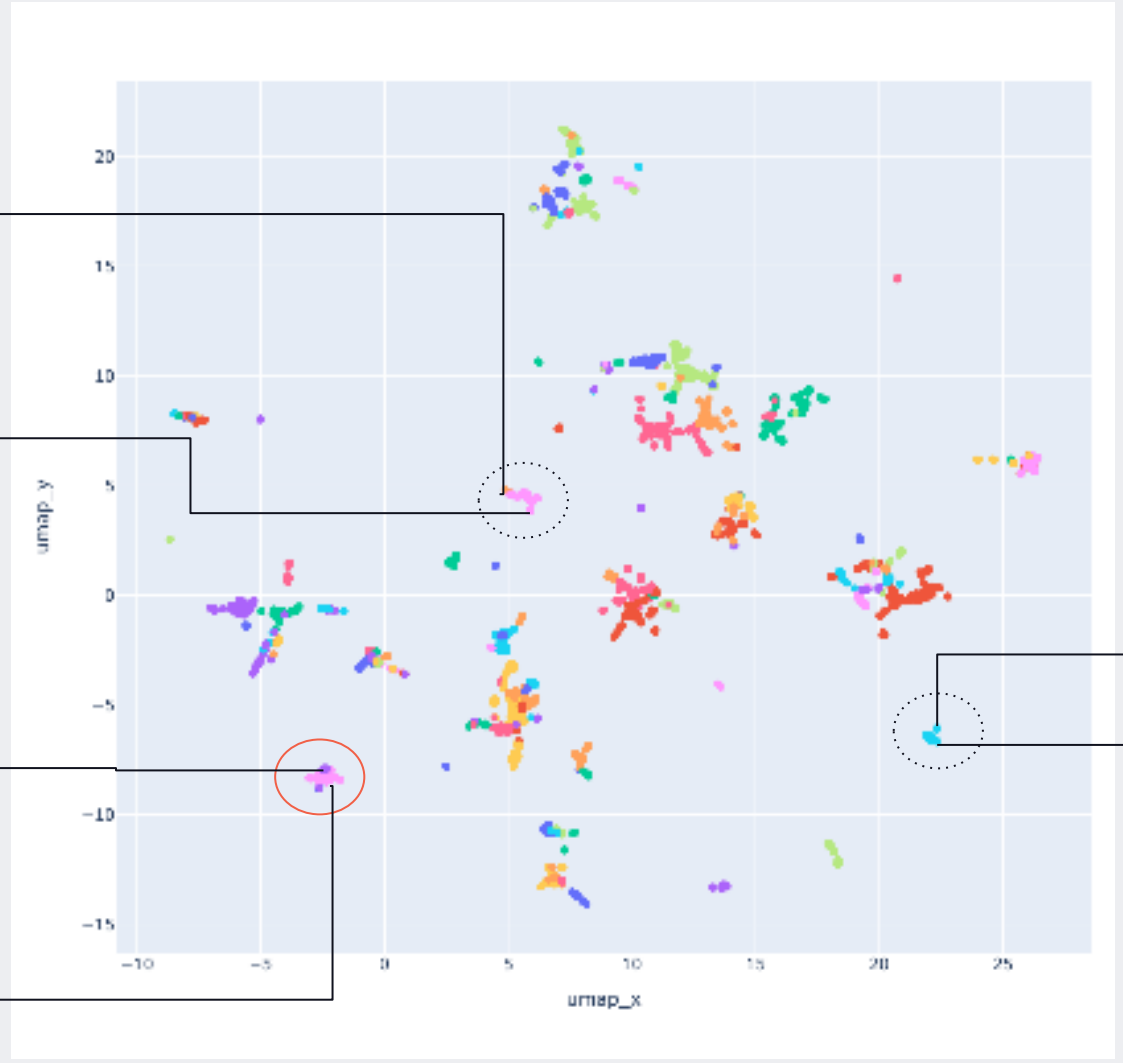
- Robustness: stable with missing input
- Multilingual support: 5 languages

What we did:

- Data augmentation with dropout
- Use of multilingual pre-trained embeddings
- Multi-task learning and hierarchical classification

Ability to compute **product2vec embeddings**

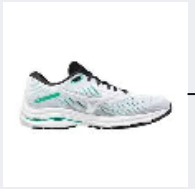
Representation of products: product2vec



Same category

Overlapping categories

Shoes > **shoes for men**



Shoes > **shoes for kids**

Same category

DATA+AI SUMMIT 2022

Products represented with product2vec + UMAP

Leveraging the product2vec

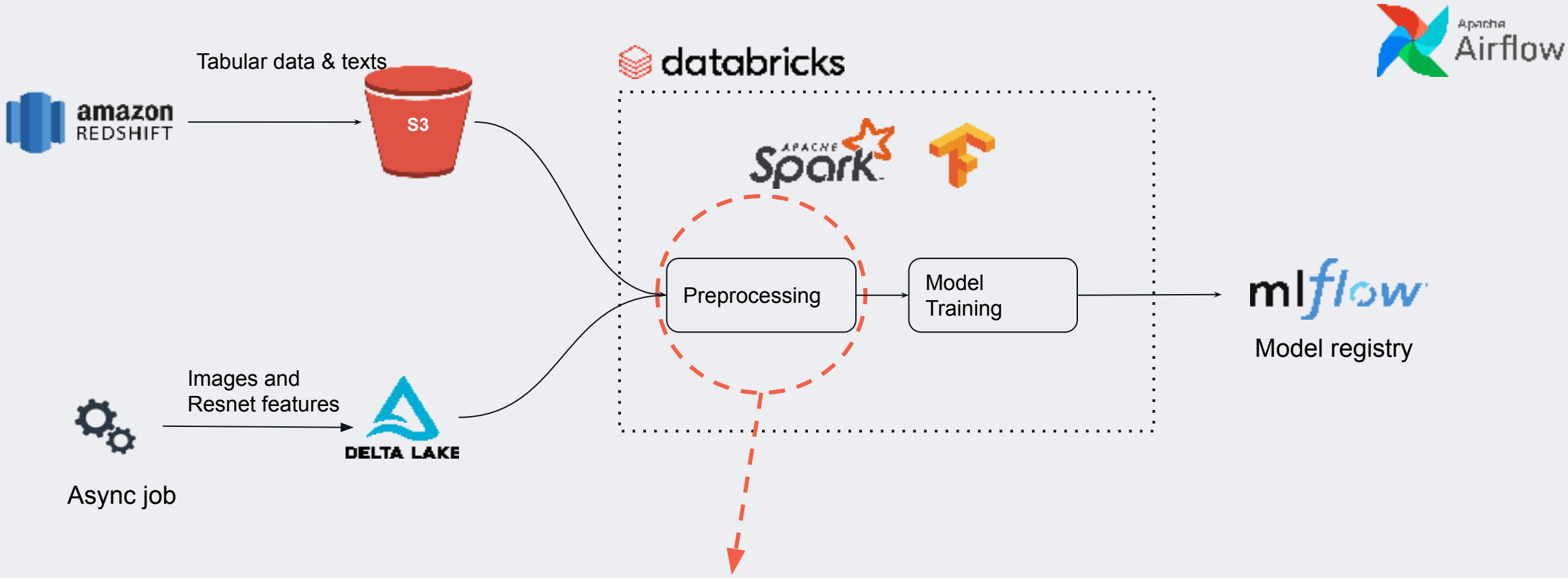
Achievements with product2vec

- **Categorization on very small catalogs**
- **Similar products detection**
- **Category embeddings**
- **Visual attribute prediction**

➢ **Use product2vec as an input for ML catalog models**

Engineering considerations

Training challenges



Training dataset creation & preprocessing:

- Distributed on multiple clusters
- Dataset size exceeding memory capacity

Training on terabytes of data using TFRecords

Why TFRecords ?

- Save texts and images as **binary files**.
- A Dataset can be composed of multiple files.
- **Allows for optimized data fetch without having to load everything in memory**.

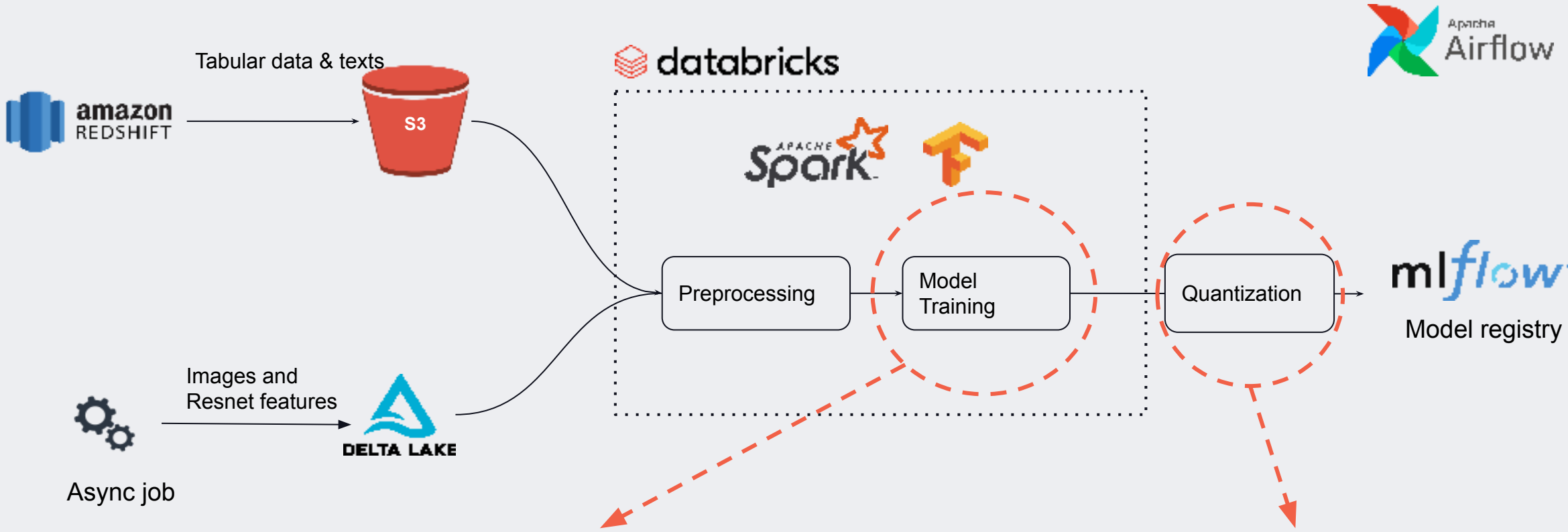

```
files = tf.data.Dataset.from_tensor_slices(filenames)


dataset = files.interleave(
        lambda x: tf.data.TFRecordDataset(x,
                compression_type="GZIP"),
    cycle_length=len(filenames),
)
dataset = dataset.map(partial(self.read_tfrecord),
        num_parallel_calls=8)

dataset = dataset.shuffle(512,
        reshuffle_each_iteration=False)

dataset = dataset.repeat()

dataset = dataset.batch(self.batch_size)

dataset = dataset.prefetch(buffer_size=100)
```

File specification and Dataset creation

Training in less than **3 hours**

- For a dataset of ~20 M products
- On a g4dn.8xlarge (single GPU)

Training challenges



Training:

- With dataset size exceeding memory capacity
- Use of multi GPU with mirrored strategy from Tensorflow

Preparation for inference:

- **Convert the model to ONNX format**

ONNX

ONNX & ONNX Runtime

ONNX - open standard format defining a common set of computational functions:

- ONNX represents deep learning models in a wide variety of frameworks
- **ONNX Runtime** provides tools to optimize the ONNX graph
- **Dynamic quantization to reduce latency and model size**

Smaller

÷ 4

Faster

-60%

With ONNX Runtime and dynamic quantization, the use of huge models like transformers (e.g. BERT models) in production is possible with CPUs

Achievements with categorization model

- **Mirakl categorizes automatically products into the right categories**

- **We obtained the cornerstone of catalog ML use cases: product2vec embeddings**

Finding duplicate products

A reliable and coherent product catalog

Finding duplicates use case

Goal : find product data from sellers that refer to the same product

Product 3029102

Name : Tennis laces man
Seller : MODE & SPORT
Brand : Bensimon
Barcode : None
Description : Opt for these canvases from the cult brand

Product 5940304

Name : BENSIMON Denim canvas
Seller : SPORT ENTERTAINMENT
Brand : Bensimon
Barcode : 3608544890070
Description : Bensimon canvas sneaker for men, model TOILE LACET, color blue

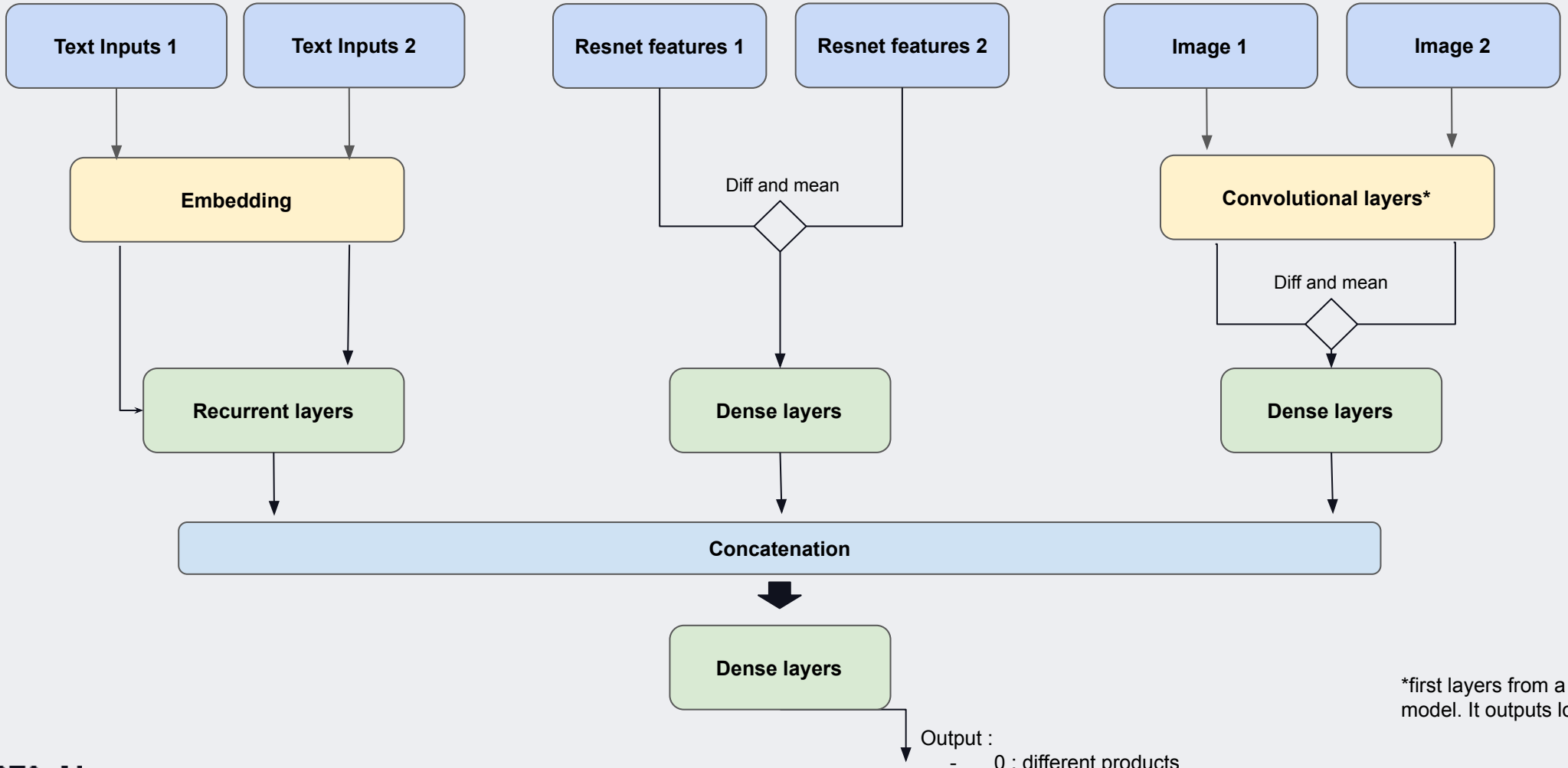
Notes:

- Catalogs may be very **dense**: **thousands** of similar products in a category
- **Sellers** may describe the same product very differently

Product2vec embeddings are not precise and robust enough for this task

Model architecture

Goal : Take both product data as input, infer whether or not they refer to the same product



*first layers from a feature extraction model. It outputs local low level features.

Focus on duplicated products search

Reference set

Product 1	Product 2	Product 3	Product 4	Product 5	Product 6	…….

✗ Cannot use product embeddings (not precise enough, either too many false positives or no positives at all)

✓ Can use an **algorithm that compares each product low-level data to tell if they relate to the same product**. Filtering and pairwise comparisons

New products set

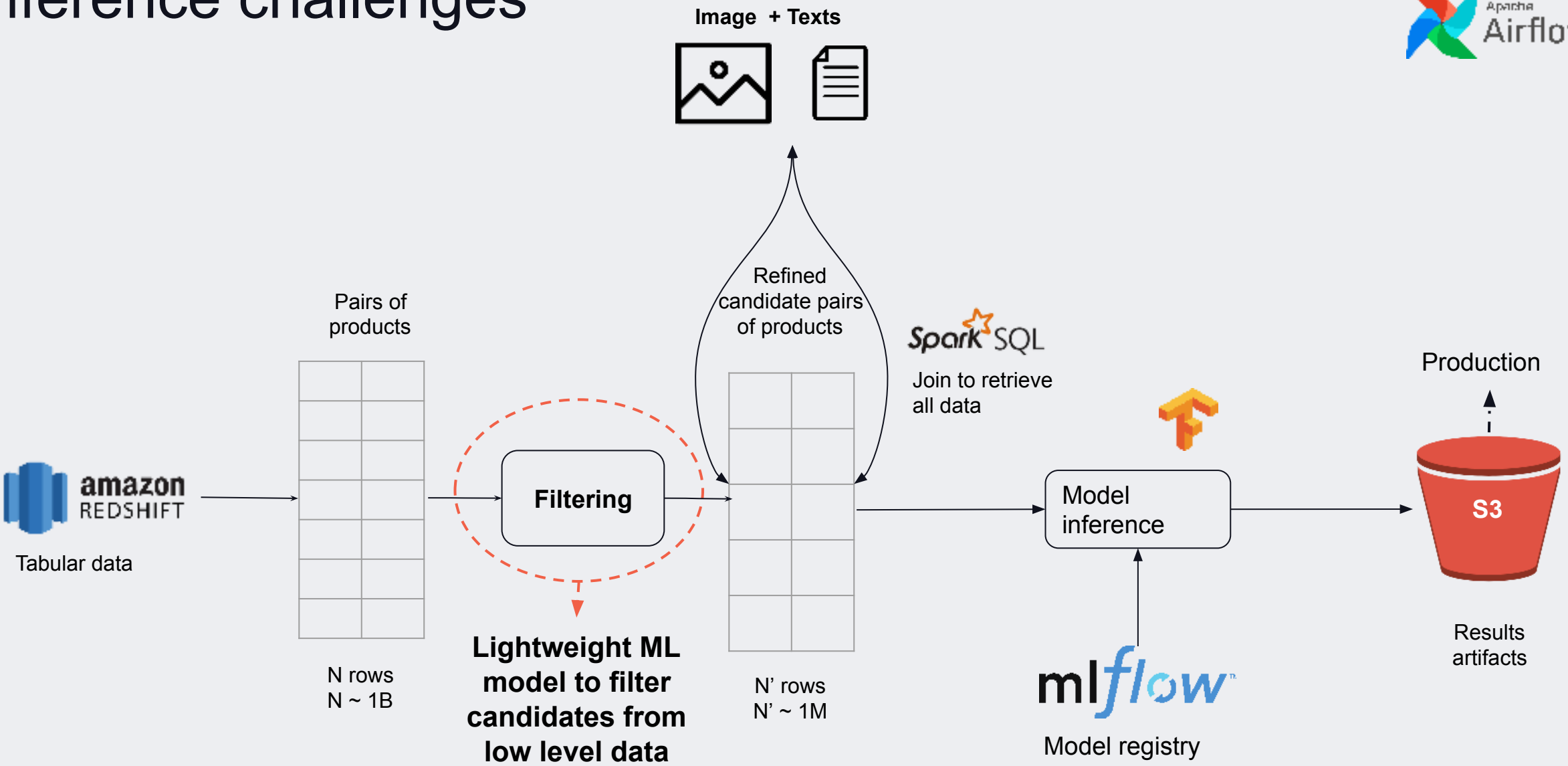
Product 1
Product 2
Product 3
Product 4
Product 5
Product 6
…….

Pairwise comparisons = O (n^2) problem

→ **We need a robust architecture**

Engineering considerations

Inference challenges

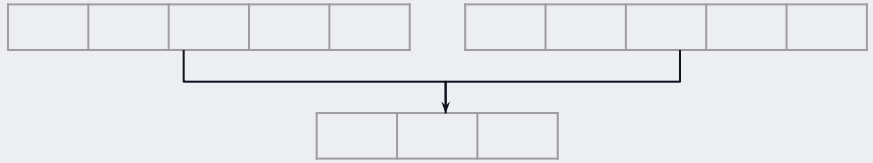


Filtering step

Goal : Drastically decrease the number of candidate pairs so that a multimodal deep learning model can be used on a limited set.

1. Compute basic features on images:
   - Histograms
   - Texture
   - Aspect ratio

2. Derive some features for pairs of images using vectorized computing

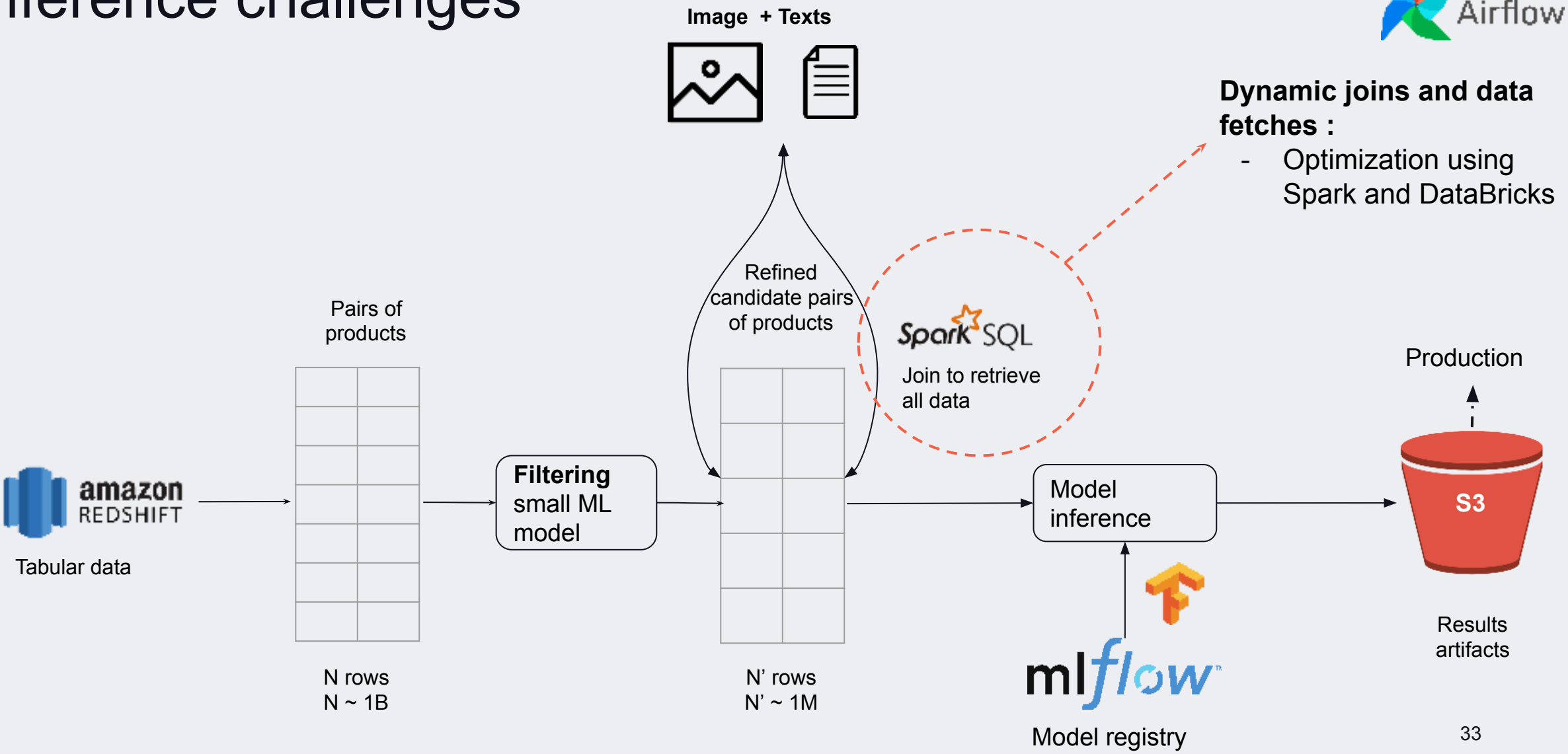


3. Apply a Decision Tree

   - 1M predictions per second

Consequence : From 1B pairs to 1M

Inference challenges



33

Spark optimization

Optimize data frames joins:

- Use of **spark.sql.autoBroadcastJoinThreshold** to tune joins or deactivate it if needed.
- Carefully **partition** files and Dataframe, and **persist** Data **On disk only** for large DataFrames (do not forget to unpersist).

Optimize data queries:

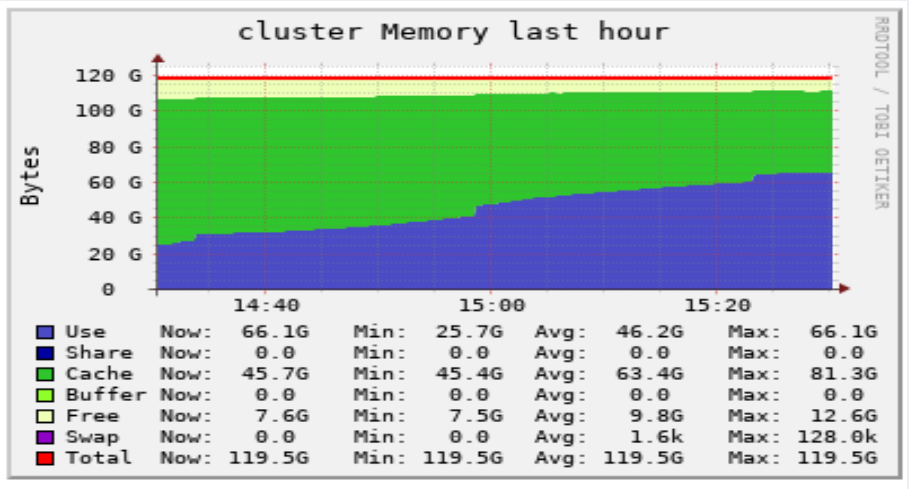
- Use of **Z-ordering** in delta tables to speed up data queries

Spark optimization

Tips

- Add a non serializable model into spark context to predict with Spark UDF/pandas UDF
- **Singleton class** avoids loading object at every UDF call and **accelerates executions**

Be careful when using custom UDF, they can introduce memory leaks




```python
from pyspark import SparkFiles

class Singleton(type):
  _instances = {}
  def __call__(cls, *args, **kwargs):
    if cls not in cls._instances:
      cls._instances[cls] = super(Singleton, cls).__call__(*args, **kwargs)
    return cls._instances[cls]

class CatalogEstimator(metaclass=Singleton):
  model = None
  def __init__(self, remote_model_path):
        File_name = get_file_name(remote_model_path)
    Estimator.model = self._load_model(SparkFiles.get(file_name))

  @staticmethod
  def _load_model(model_path):
    return catalog_model.load_model(model_path)

  @staticmethod
  def predict(remote_model_path, sentence):
```

```python
spark.sparkContext.addFile("/tmp/model.bin")
@pandas_udf(T.StringType())
def predict_pandas_udf(s: pd.Series) -> pd.Series:
  s = s.apply(lambda x: CatalogEstimator.predict("/tmp/model.bin", x))
  return s
spark_DF.select('id', predict_pandas_udf(col("body")).display())
```

Conclusion - key takeaways

- **Multimodal product embeddings are the cornerstone of Machine Learning on Catalog Data**
- **Before pulling out the big guns, check if pragmatic and frugal solutions can resolve your $O(n^2)$ problems**
- **Optimization on Spark and inference time reduction with ONNX are paramount to scale your pipeline**

DATA+AI
SUMMIT 2022

Thank you