DATA+AI
SUMMIT 2022

Implementing an End-to-End Demand Forecasting Solution Through Databricks and MLflow

Ivana Pejeva
Cloud Solution Architect, Microsoft

Yoshi Coppens
Data Engineer, element61

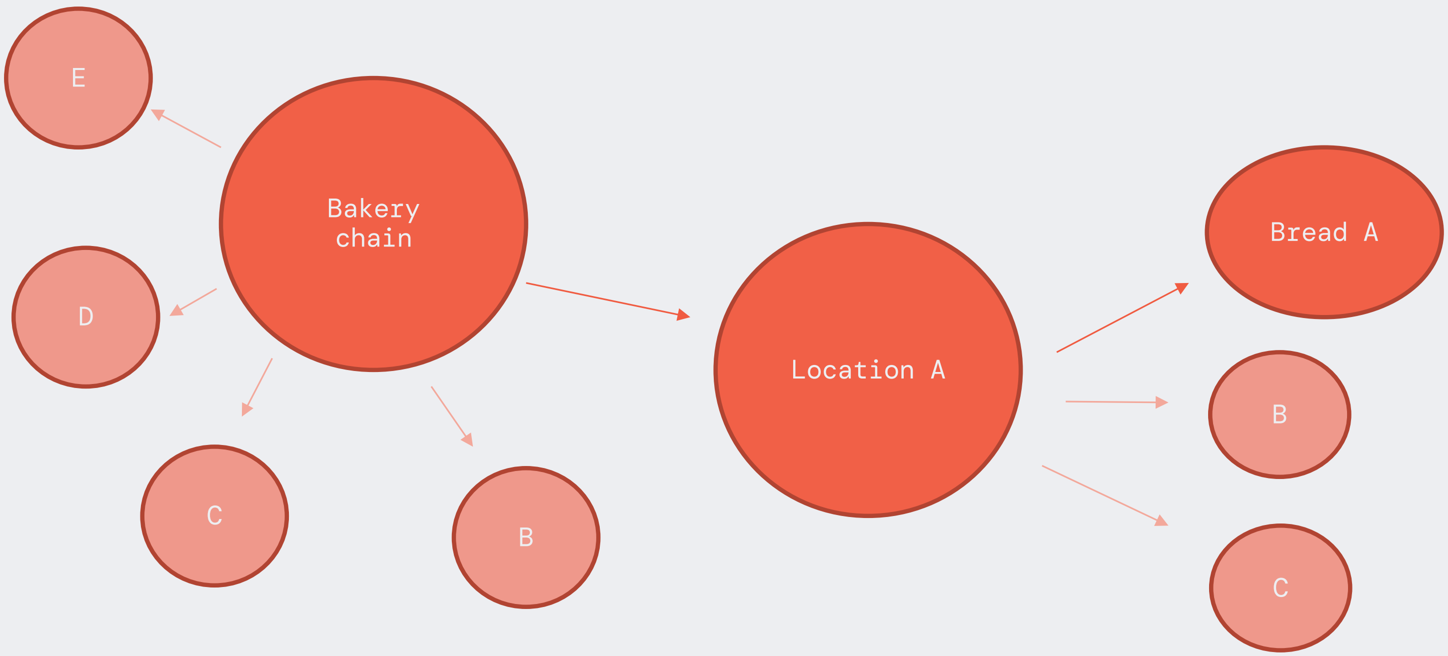
ORGANIZED BY databricks

Outline

- Problem Setting
- Ingestion
- Cleaning, transforming, enriching
- Model selection and training
- Scoring
- Feeding it back to the client's system
- Running the model in production

Problem setting

Scope of Project



How much to sell of Bread A in
Location A

DATA+AI
SUMMIT 2022

Ingestion

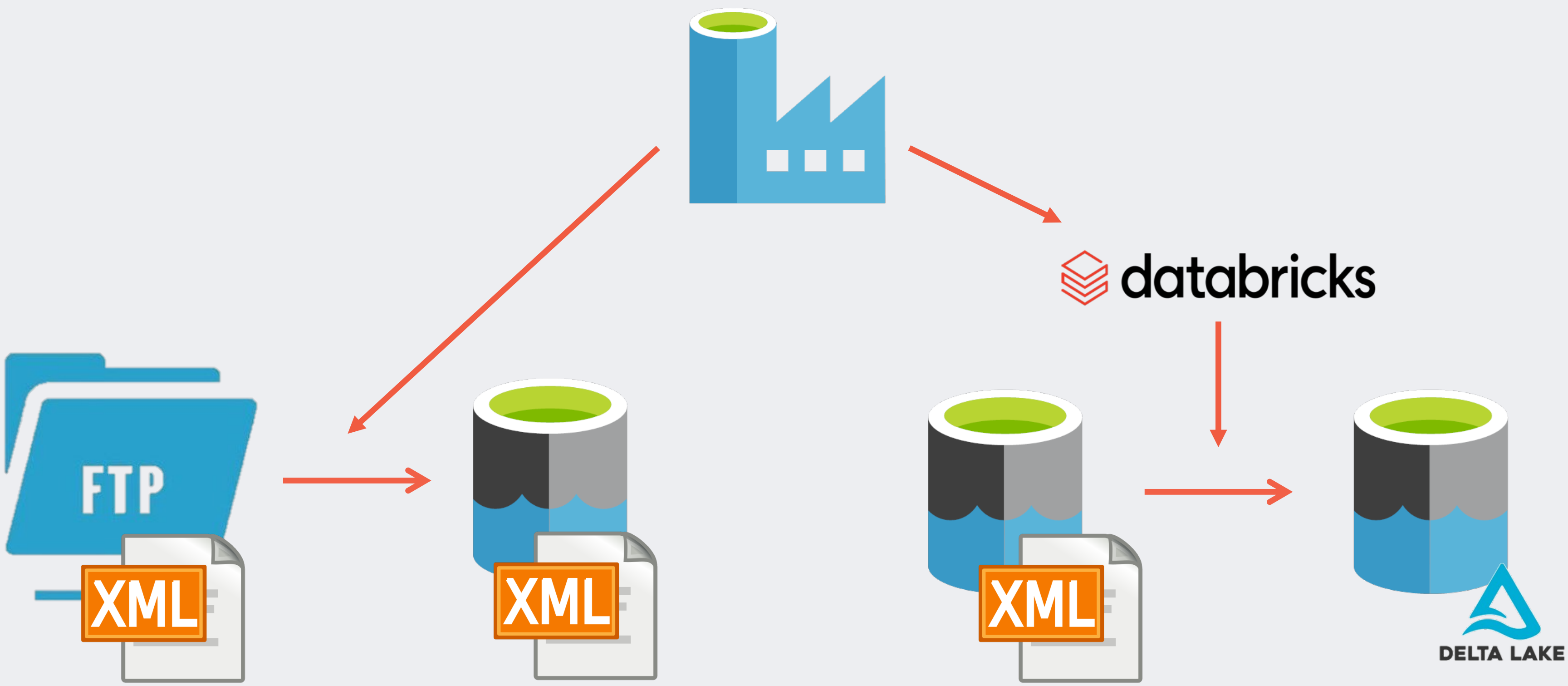
Ingesting the files

XML files on an FTP server??

- The provided salesdata is made available every day on an FTP-server
- File format is XML
- Azure Data Factory used to copy the binary files from FTP server
- Use spark-xml package in Databricks to translate from XML
  https://mvnrepository.com/artifact/com.databricks/spark-xml_2.12

Ingesting the files

Structure of Data Lake and ingestion



Ingesting the files

Learnings

Ideally the data gets fed into your systems in an optimal way but when you cannot choose the setup, you have to make it work, ideally not having to incorporate more and more different tools

Combining Azure Data Factory with Azure Databricks gives you quite a lot of power

Cleaning, transforming, enriching

Using extra data sources

Client based

Store Information

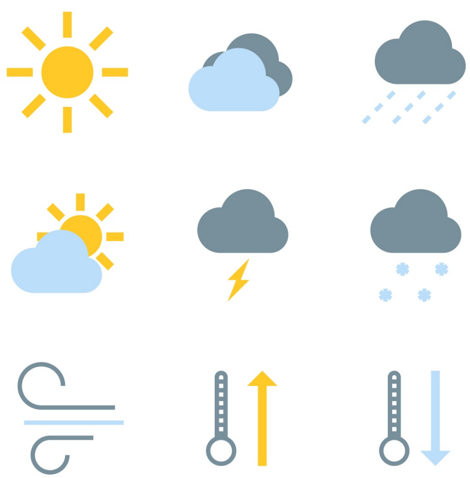
Product Information

Store Holiday Information

Competitor Holiday Information

Using extra data sources

External sources



Feature Engineering

- Feedback from client is important
  - ‘felt that model was too slow with incorporating a big shift in demand trend’
  - incorporate **Sales of last week**
- Analyzing performance is important
  - ‘we saw that there seemed to be a bias to underestimate usually’
  - when they sell 10 but only had an inventory of 10 vs selling 10 when there is 20 delivered, huge difference => incorporating a **SoldOut** factor

Model Selection and Training

Picking a ML model

Learnings

ARIMA vs ML

- Very dependent on data and problem when one is performing better than the other
- Big advantage in ML => make a joint model from multiple time series

Picking LightGBM

- Focus was not on complicated ML due to time constraints
- We wanted some level of **interpretability** for the client

Model Granularity

1 Model for all stores

- Features such as Adoption Level, Location, Population, Events etc.
- We did not get the required information

1 Model for all products

- Difficult, pain au chocolat vs croissant vs éclair?

1 Model per product group

- Yes, can distinguish some fungible groups, such as Bread, Confisserie, Small Cakes

Making use of parallelism

Training multiple models at once using Pandas UDFs

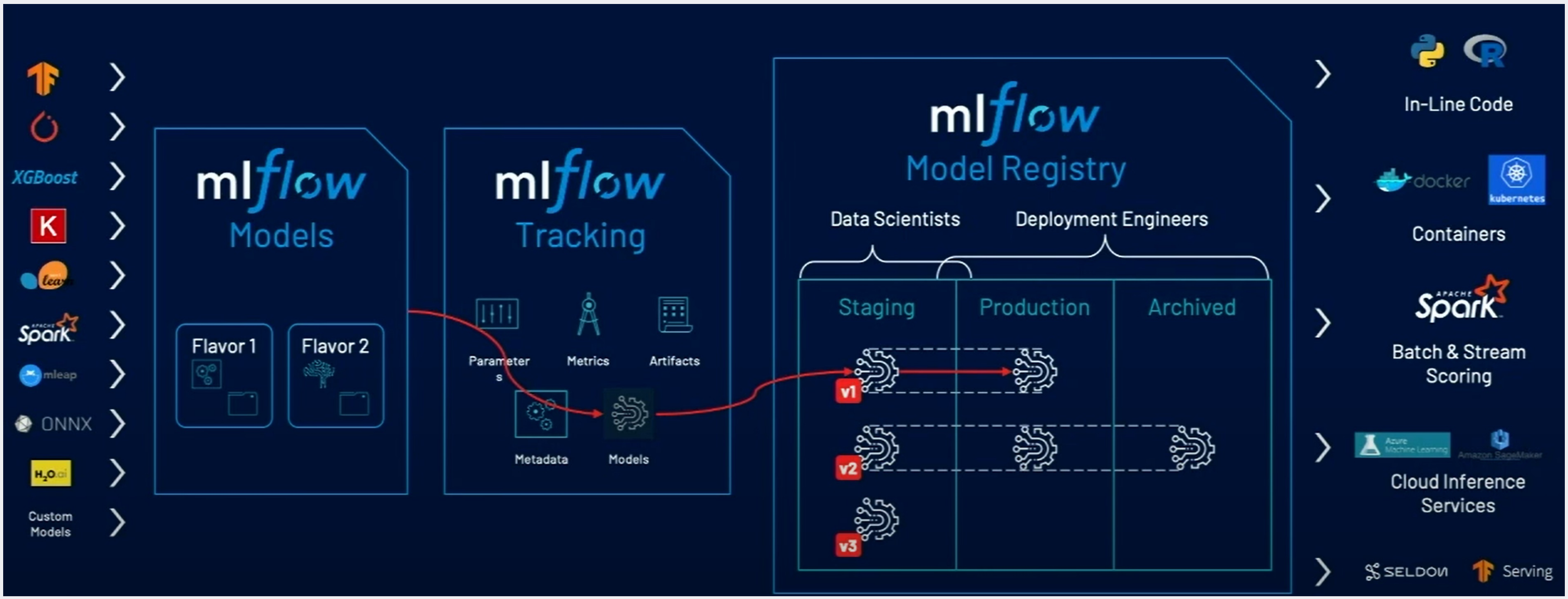
- We have multiple productgroups and multiple stores => **248 models**
- Spark MLLib is not made for training multiple models at once
- A for loop of course is also very unperformant

- **Grouped Map Pandas UDFs! For both Training and Scoring!**

```
df
.groupBy(['StoreKey', 'ProductGroupKey'])
.applyInPandas(trainingUDF, schema = schema_training)
```

Tracking Performance and experiments

Introducing MLFlow



Source: Spark AI Summit - Richard Zang & Denny Lee

Tracking Performance and experiments

Continued MLFlow - code for training

```
def trainUdf (pandasDf:object):
    ...
    with mlflow.start_run(experiment_id=experiment_id):
        gbm = lgb.train(...)
        X_test['pred'] = gbm.predict(...)
        ...
        mlflow.log_metric("MAPE", MAPE)
        ...
        mlflow.lightgbm.log_model(lgb_model = gbm,
        artifact_path=f"{storeKey}_{productGroupKey}_model",
        registered_model_name=f"{storeKey}_{productGroupKey}_model")
    ...
```

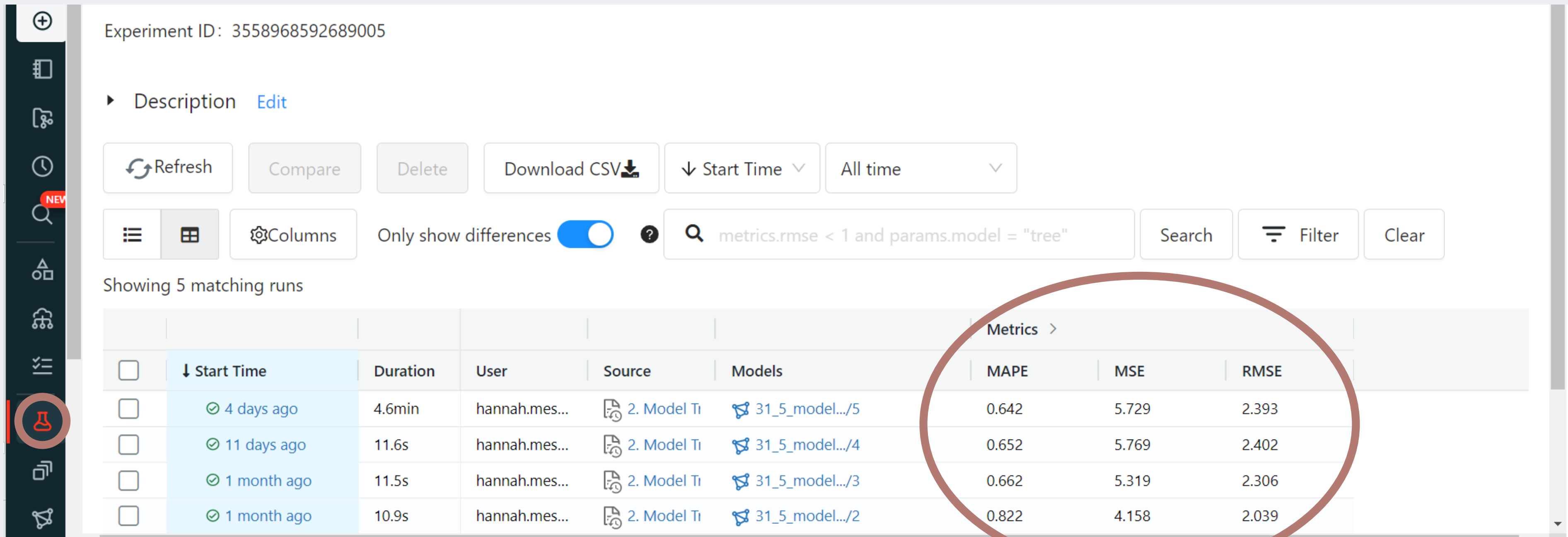
Tracking Performance and experiments

Continued MLFlow - code for scoring

```
def predictUdf (pandasDf:object):
    ...
    model_name = f"{storeKey}_{productGroupKey}_model"
    model_uri = f"models:/{model_name}/Production"
    model = mlflow.lightgbm.load_model(model_uri)
    predictions = model.predict(...)
    ...
```

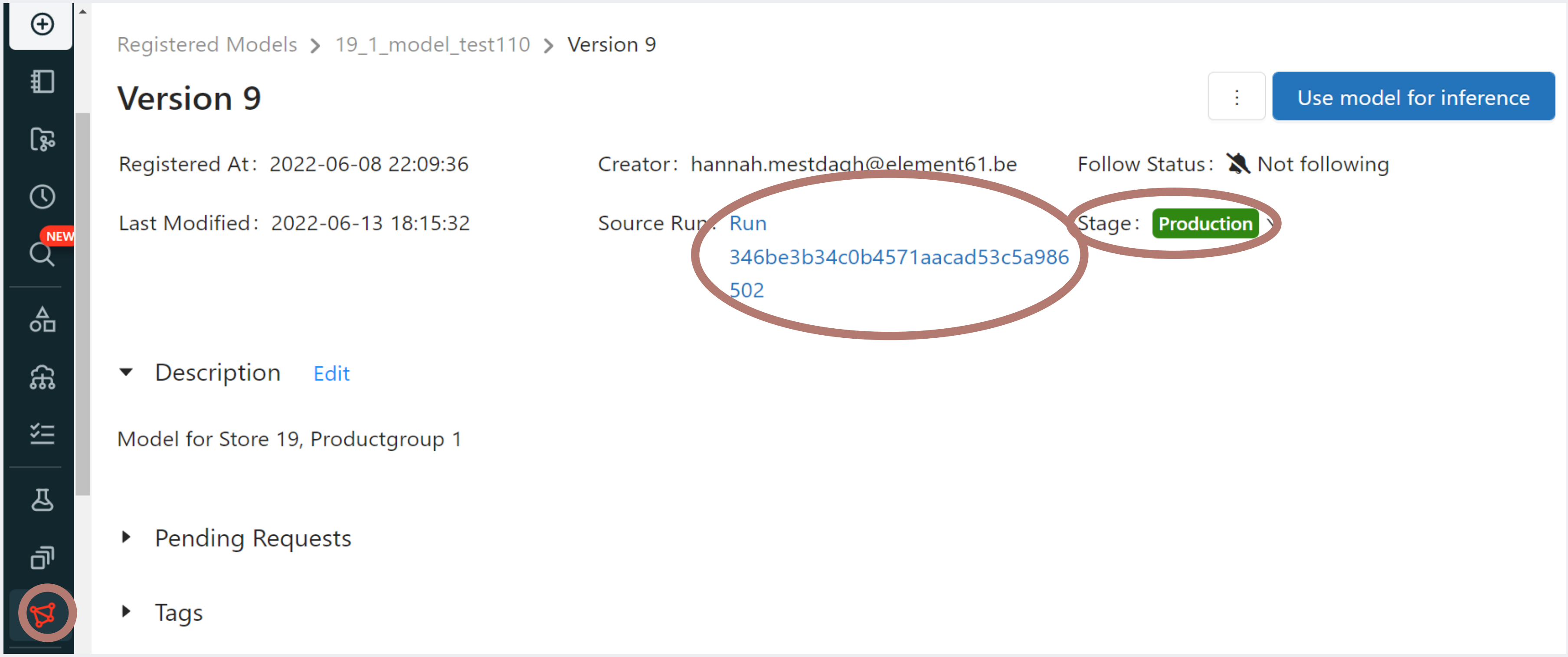
Tracking Performance and experiments

Databricks - Experiments



Tracking Performance and experiments

Databricks - Models



Scoring

Which metrics to use?

Can’t go wrong with MAPE

- MAPE = Mean Absolute Percentage Error
- Discussion with client: Overestimating is better than underestimating!
  - Option 1: Change our metric to punish missing sales harder than having leftovers (using production cost and opportunity loss)
  - **Option 2: Instead of using the quantity sold, use quantity + wanted margin**

Working with reliability buckets

- We want to know for which products we can have a reliable predictions

What can we predict well

What is okay

What is not necessarily great

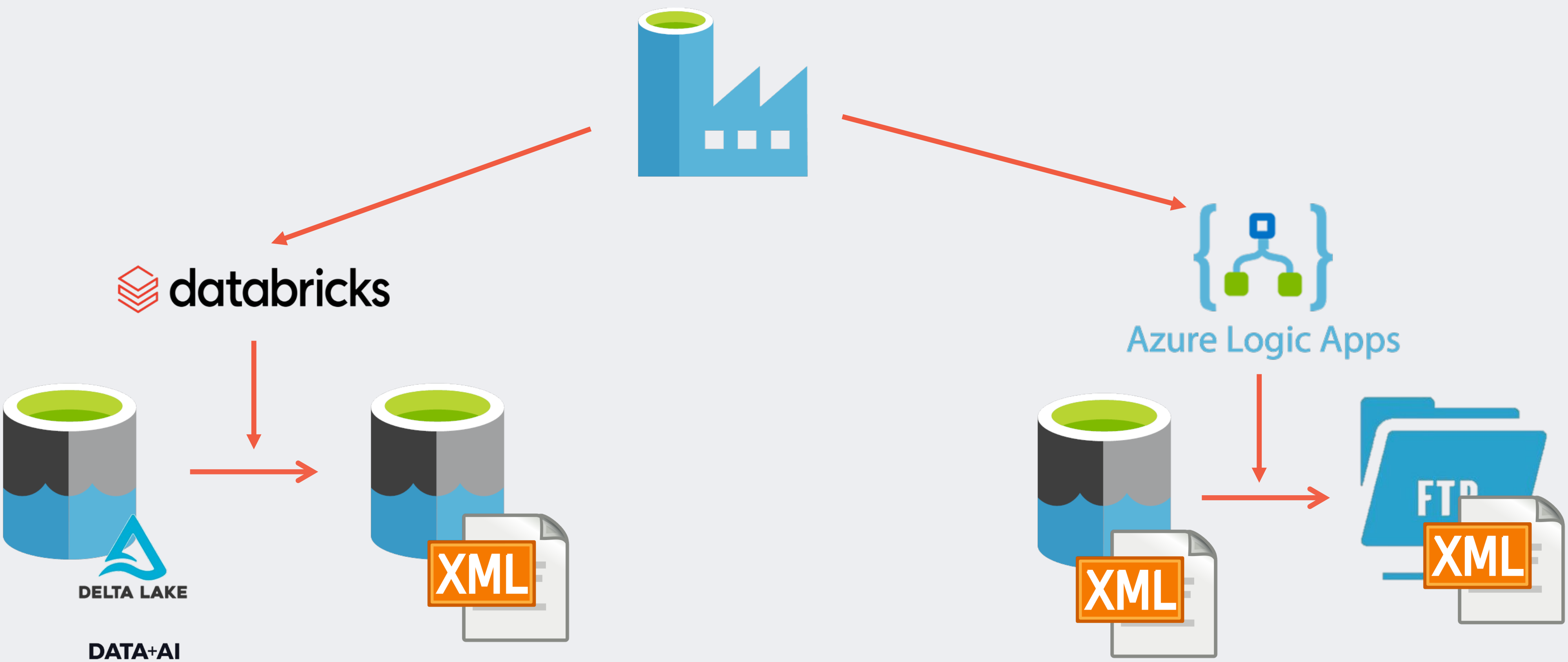
Feeding it back into the client's systems

Feeding it back into the client's systems

XML and FTP again!

- Their ERP system needs XML files again!
  - Translate Delta Lake tables back to XML using spark-xml package in Databricks
- XML files need to be transferred back to the FTP server
  - Azure Data Factory actually cannot do this "out-of-the-box", can only copy to SFTP
  - Introduce **Azure Logic App** which can use FTP as a sync and has a lot more data connectors

Feeding it back into the client's systems



Running the model in production

Frequency of training and scoring

Multiple factors matter

Data can be weeks late

- Internal system can have big delays
- Forecast is needed every day
- Weather has a big impact so scoring every day!

Keep control of cost

- The cost of training is not negligible
- Rather more frequent scoring than more frequent training

Importance of new data

- For old stores, new data gets less and less important
- For new stores, it is the opposite
- They are expanding and for consistency's sake training every week!

Monitoring and Alerting

Mailing through Logic App, orchestrated in Data Factory

Data missing

- Due to operational difficulties, extensive checks needed => Mail
- Client input data not getting updated => Mail
- Follow up for new stores being added

Monitoring models

- Big improvement in MAPE => Mail
- Deteriorating MAPE vs last => Mail
- Future: Model Drift Detection

Conclusion

Conclusion

- In a couple of days we can get a Demand Forecasting model into production
- **Pandas UDFs can help us scale** hugely in the model training
- **MLFlow is a great MLOps tool** for tracking, registering Machine Learning models

DATA+AI SUMMIT 2022

Thank you

Ivana Pejeva & Yoshi Coppens