

Building your Intelligent Lakehouse

by Git Kraken



Final Group Project - CSCI 103

Data Architect - Muhammad Sebuliba

Data Architect - Josh Laka

Data Engineer - Richard Kwan

Data Engineer - Kirk McGraw

Data Scientist - Jennifer Hu

Data Scientist - Hellen Momoh

BI Analyst - Kenan Touma

Coordinator - Richard Kwan



Agenda:

- Roles
- Problem Statement
- Data Architecture
- Data Engineering
- Data Science
- BI Analyst
- Insights and Business Recommendations
- Planned Technology Refinements





Business Use Case

Problem Statement - *To predict seasonality and other external effects on overall sales and performance*

Grocery stores struggle with forecasting sales

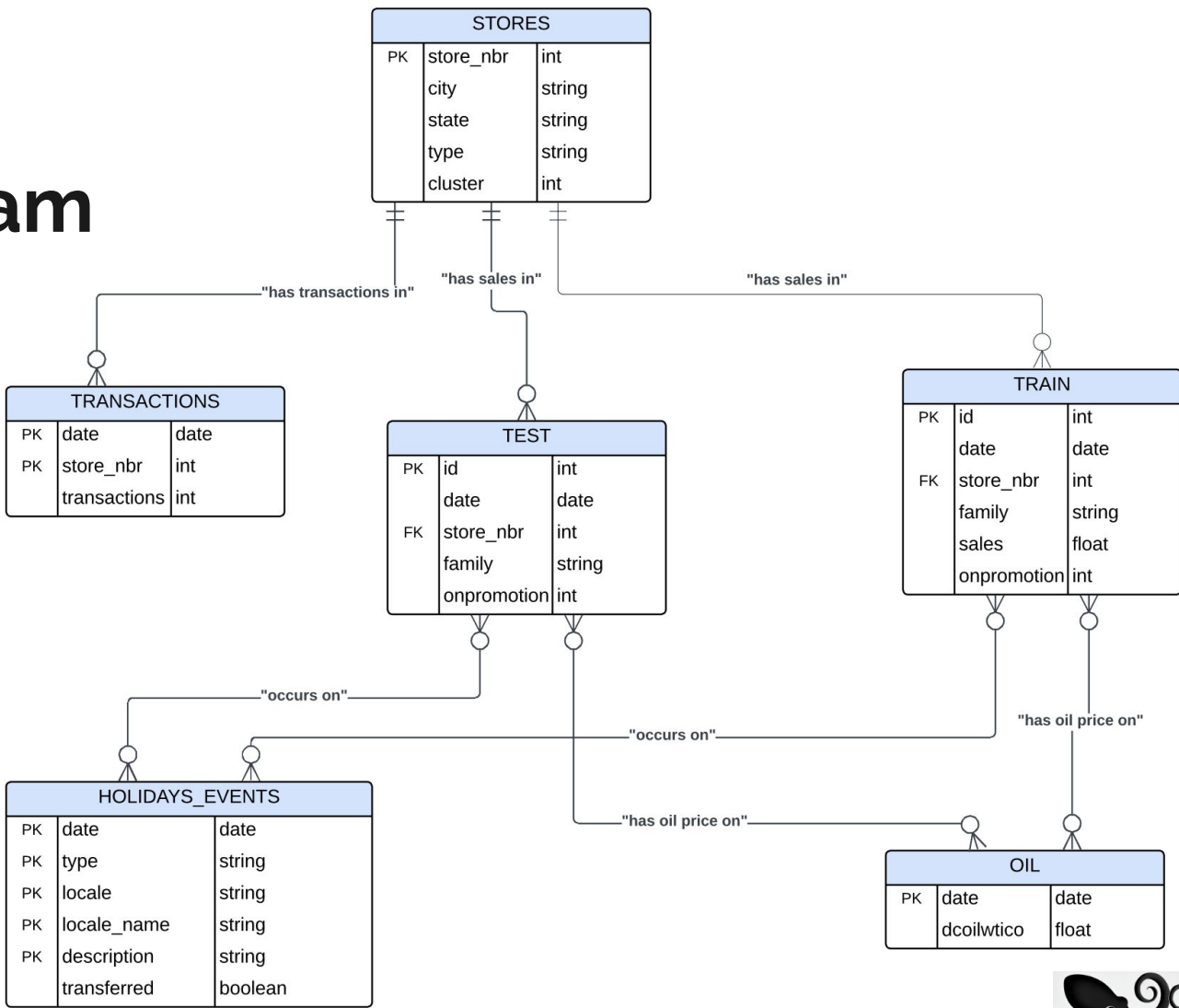
- Forecast is too low → Store is consistently low on stock → Poor customer satisfaction and the store misses out on sales
- Forecast is too high → Perishable goods go to waste → High cost to the business





Entity Relationship Diagram

- **Indexes:**
 - Train set - id
 - Transactions - date
 - Stores - store_nbr
- **Cardinality**
 - Stores:** One-to-Many with Transactions, Train, Test
 - Train:** Many-to-Many with Holiday_Events and Oil
 - Test:** Many-to-Many with Holiday_Events and Oil





Impact & Performance of Partitions

- Final model only trains on data after June 2015
- Partition: Train 1,373,084 and Validate on 26,730

Impact of partition:

- Data dated too far back can negatively impact model
 - Specifically during earthquake skewed data
- Improves model efficiency

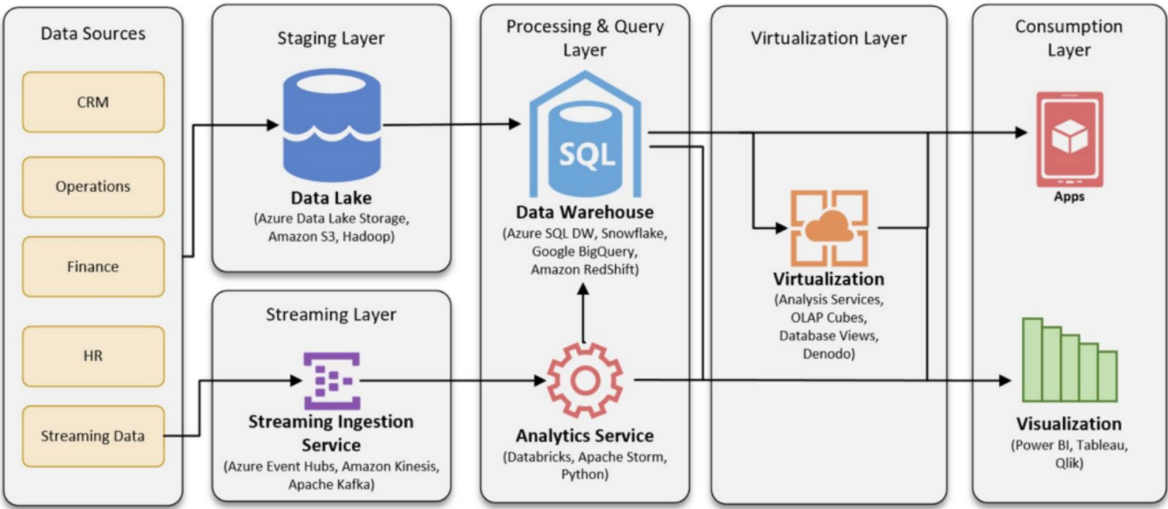




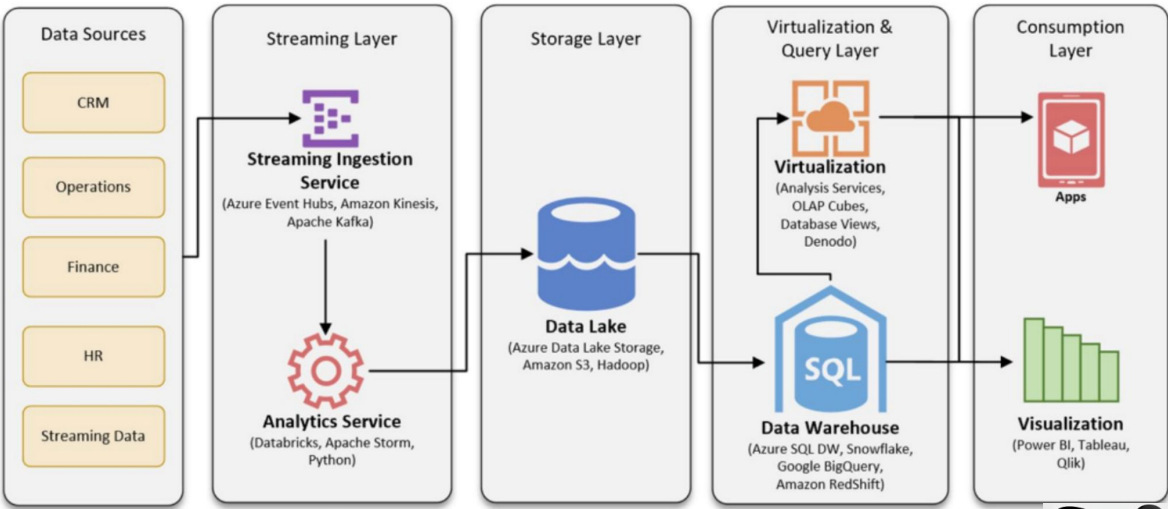
Streaming Solutions

Purpose: To provide real-time data analytics on the evolving transactions & sales dataset

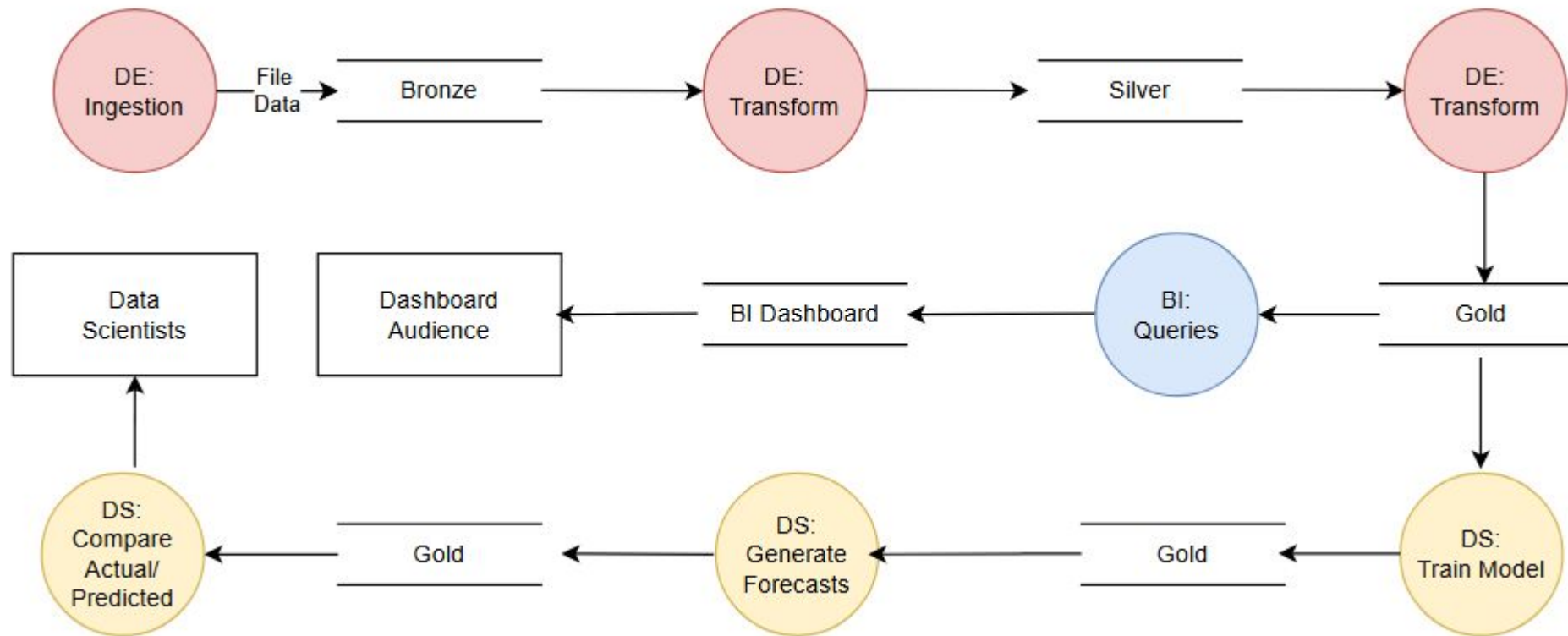
LAMBDA Architecture



KAPPA Architecture



Refined Data Flow Diagram





CI/CD Deployment Process

CI/CD Process for Deploying Code

A **CI/CD (Continuous Integration / Continuous Deployment)** process ensures that new changes are integrated, tested, and deployed automatically, minimizing errors and speeding up development

Area	Process	Tools Used
Code Backup	GitHub Repo, Feature Branches, Rollback via Git Commands	GitHub, GitHub Actions
Code DR	Used Databricks Repos for notebook versioning	Databricks Repos, Git
Data Backup	Delta Lake Time Travel, Cross-region Backups	Delta Lake, Azure Blob
Data Recovery	Rollback to Delta Lake snapshot	Delta Lake SELECT VERSION
Code Deployment	GitHub to Databricks CI/CD Pipeline	GitHub Actions, Jenkins





Data Engineering - Ingestion

- Ingestion from disk storage into bronze tables
- Silver transformations for data cleanup, feature engineering
- Gold for BI reporting and model training
- Streaming pipeline for daily BI updates using availablenow trigger due to trigger once deprecation.





Transformations/Feature Engineering

- Major Transformations
 - Reduced holiday data into boolean IsWorkDay
 - Carried over oil prices into the weekend so pricing is continuous
- Major Features created:
 - 7D/28D Moving Averages
 - Day of Week/Month 15/Month End
 - MoM/YoY Growth Rate
 - Promotion Lift Factor
 - Seasonal Index
 - Market Share
 - Sales Growth Rate
 - Average Store Sales Performance
 - Holiday Impact Factor
 - Oil Price Sensitivity



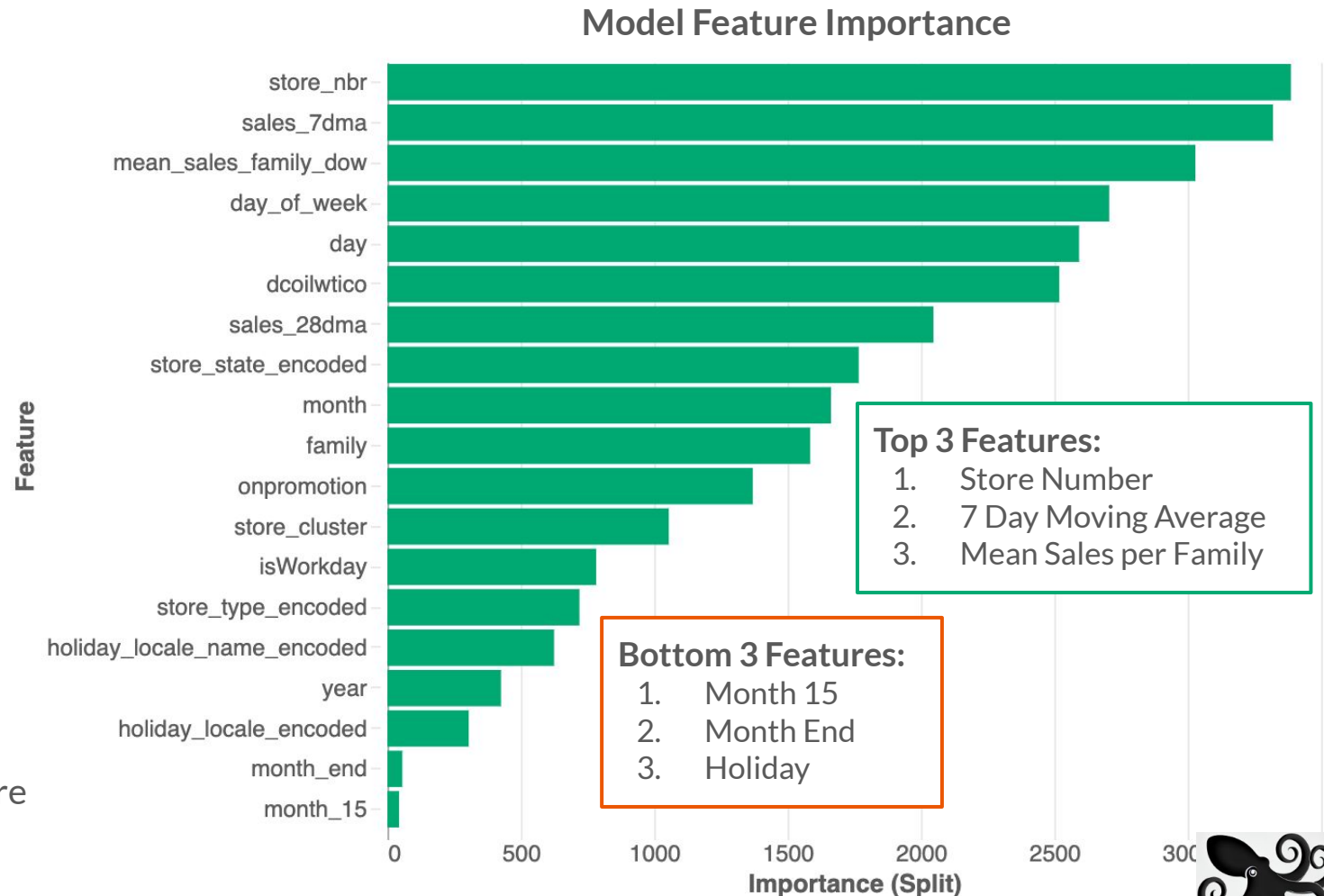
Going Gold and Operationalizing

- DE Gold Outputs - for further joins and refinement by DS / BI
 - ◆ Train, Test
 - ◆ Store Performance, Family Performance, Product Analytics, Regional Metrics
- Operationalizing
 - ◆ Catalog permissioning
 - ◆ Pipeline:
 - Merge updates
 - Logging, error handling and retries
 - Data validation
 - Performance metrics
 - Monitoring job runs in Databricks UI
 - Setup alerts for failures
 - Track execution metrics
 - Reviewing logs

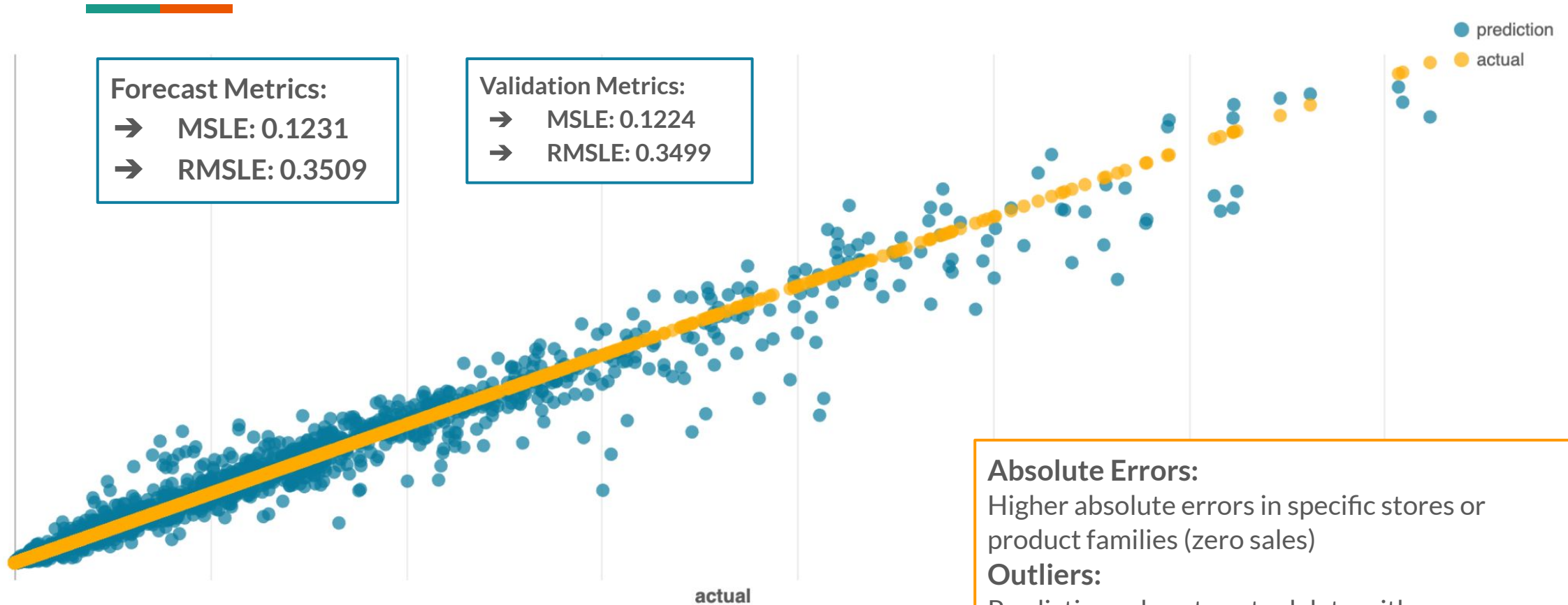


Data Processing and Modeling

- Data processing:
 - ◆ Null, duplicates, pandas
 - ◆ Split into train, validation test
 - ◆ 19 features
- Modeling: Light GBM Model
 - ◆ boosting: gbd
 - ◆ type: regression with log-link
 - ◆ number of iterations: 1000
 - ◆ Model RMSE: 179.25
- Mlflow/Pipeline Integration
 - ◆ Mlflow for model registration, version and life cycle
 - ◆ logging parameters, datasets, metrics, and artifacts
- Predictions
 - ◆ last 15 days of data for each family per store
 - ◆ run daily with incoming data



Model Forecasts vs Actuals



Absolute Errors:

Higher absolute errors in specific stores or product families (zero sales)

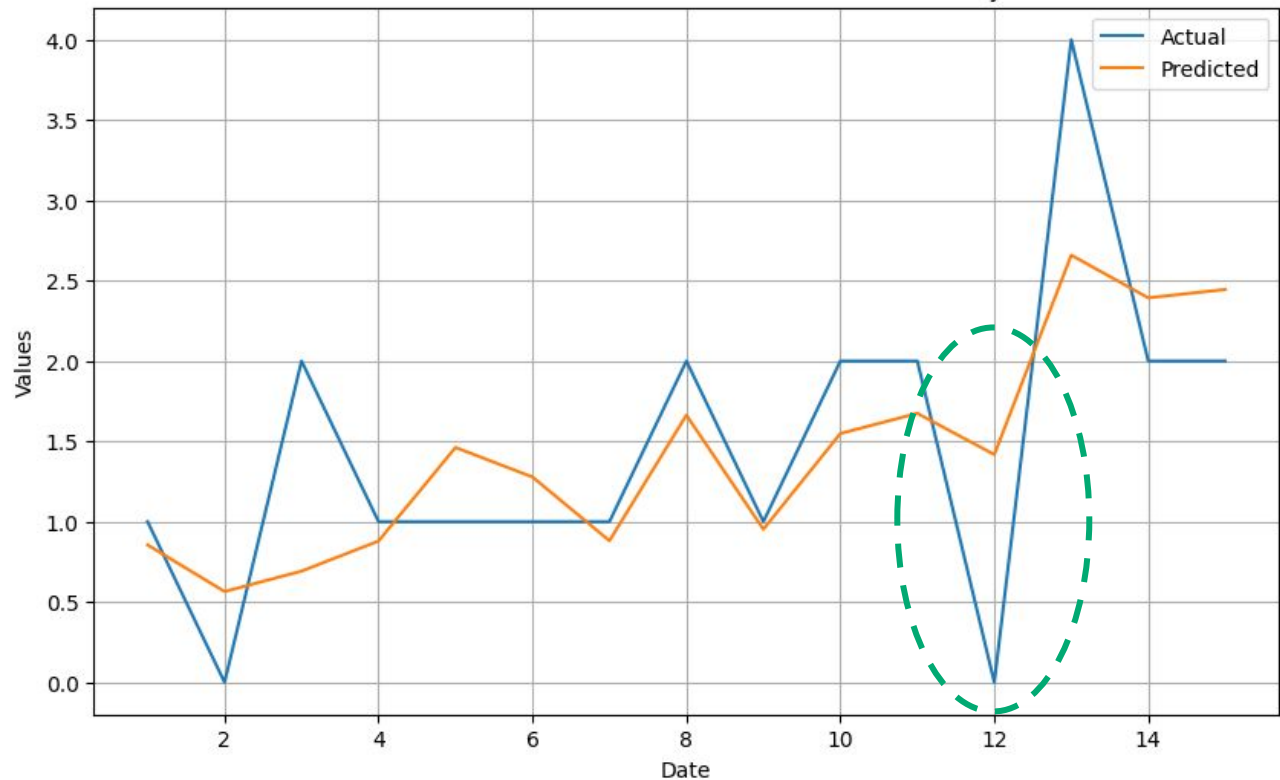
Outliers:

Predictions close to actual data with some outliers.

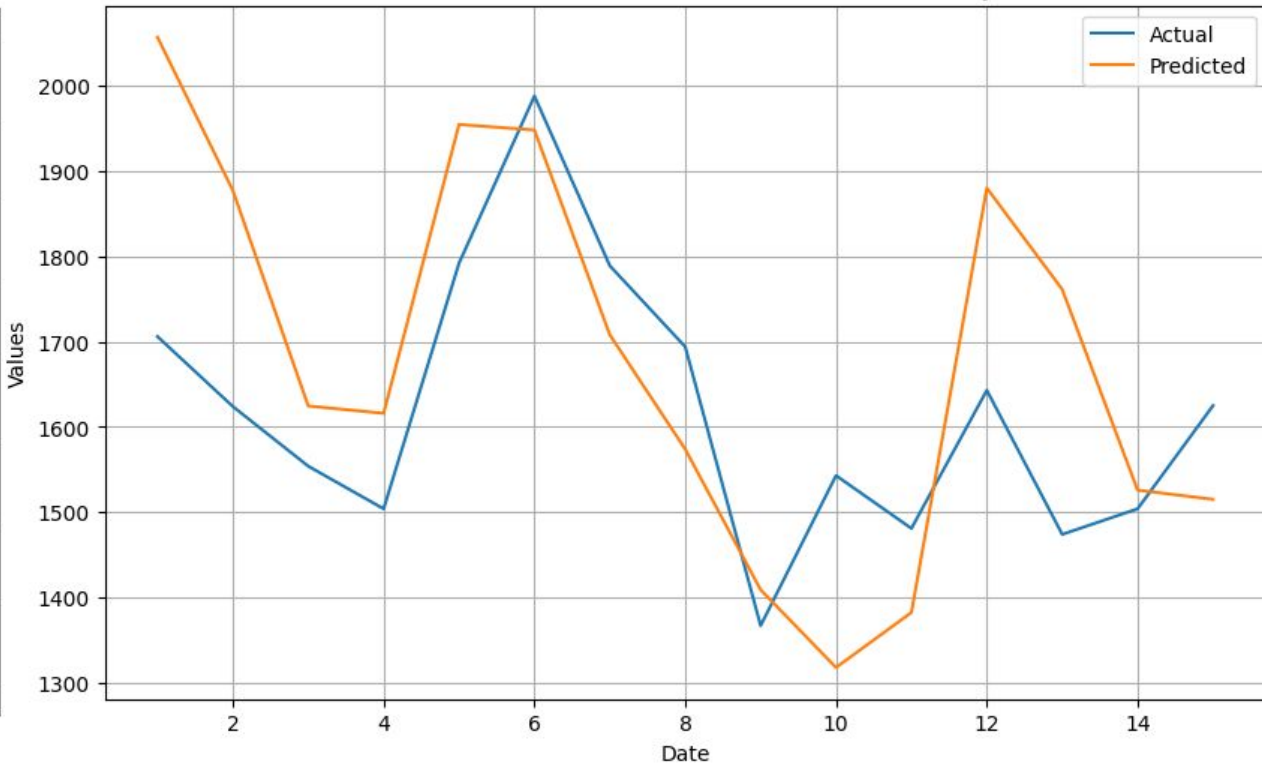


Model Forecasts

Actual vs Predicted for Store Number 12 and Family 3



Actual vs Predicted for Store Number 12 and Family 4



Sales Predictions for specific Store number and Family → follows actual sales for the most part

Improvements: Optimize feature selection, model tuning to better predict volatile sales data



BI Analyst

BI Dashboards - Sample SQL Queries

```
1 SELECT
2     month,
3     SUM(monthly_sales) AS total_sales
4 FROM
5     hive_metastore.cscie103finalprojectgroup1fall2024.gold_family_performance
6 WHERE
7     year < 2017
8 GROUP BY
9     month
10 ORDER BY
11     1 ASC
```

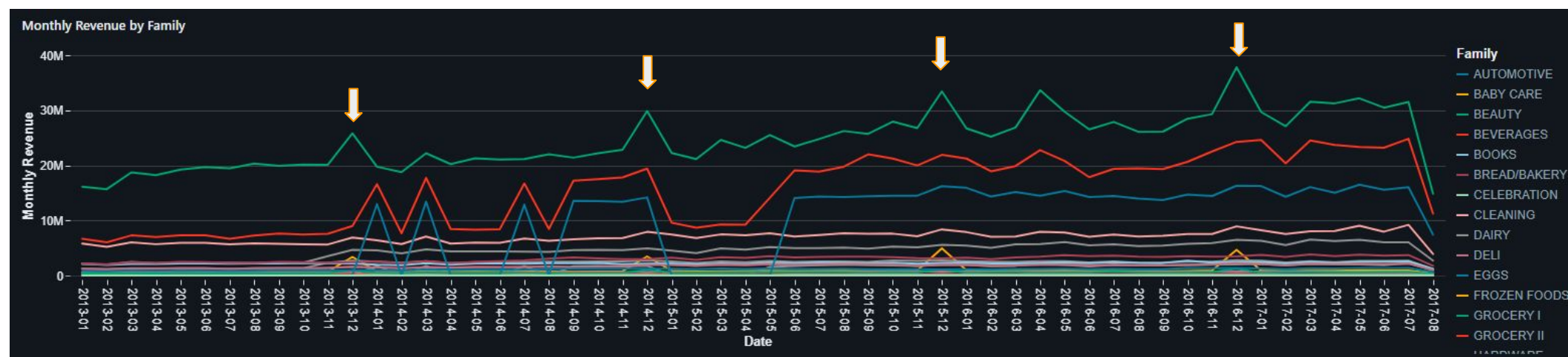
```
1 SELECT
2     family,
3     CONCAT(year, '-', month) AS monthyear,
4     SUM(total_revenue) AS monthly_revenue
5 FROM
6     hive_metastore.cscie103finalprojectgroup1fall2024.gold_product_analytics
7 GROUP BY
8     family
9     , year
10    , month
11 ORDER BY
12     family
13     , year
14     , month
```

```
1 WITH TopStores AS (
2     SELECT
3         store_nbr,
4         SUM(total_sales) AS total_sales
5     FROM
6         hive_metastore.cscie103finalprojectgroup1fall2024.gold_store_performance
7     GROUP BY
8         store_nbr
9     ORDER BY
10        total_sales DESC
11     LIMIT
12        5
13 )
14
15 SELECT
16     sp.store_nbr,
17     CONCAT(sp.year, '-', sp.month) AS monthyear,
18     SUM(sp.total_sales) AS monthly_sales
19 FROM
20     hive_metastore.cscie103finalprojectgroup1fall2024.gold_store_performance AS sp
21 JOIN
22     TopStores AS ts
23 ON
24     sp.store_nbr = ts.store_nbr
25 GROUP BY
26     sp.store_nbr,
27     sp.year,
28     sp.month
29 ORDER BY
30     sp.store_nbr,
31     sp.year,
32     sp.month
```

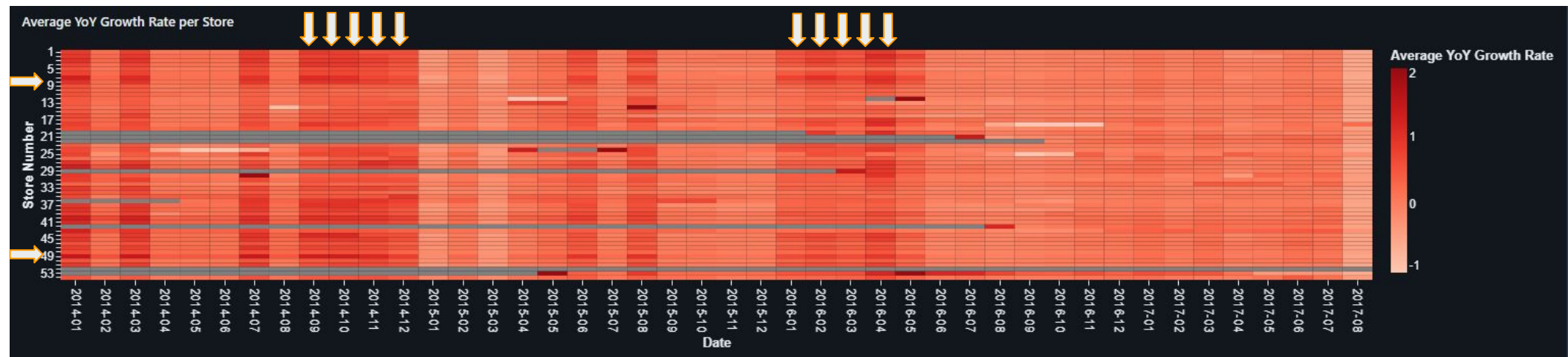


BI Analyst

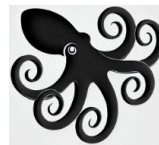
BI Dashboards - Seasonality



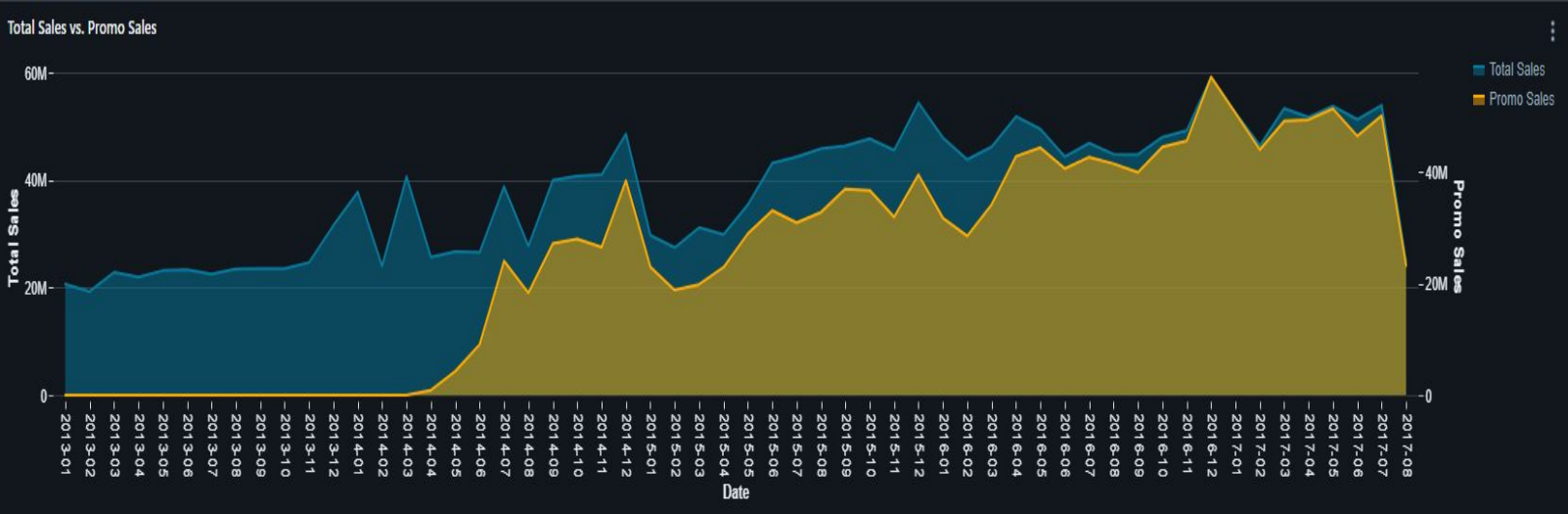
We notice monthly peaks every December of every year. This is indicative of a seasonal trends where end of year / holiday season drives customers through the doors.



YoY Growth rate seems to be less affected by seasonality. Potentially driven by decisions from upper management / overall economic standing.



BI Dashboards - External Factors



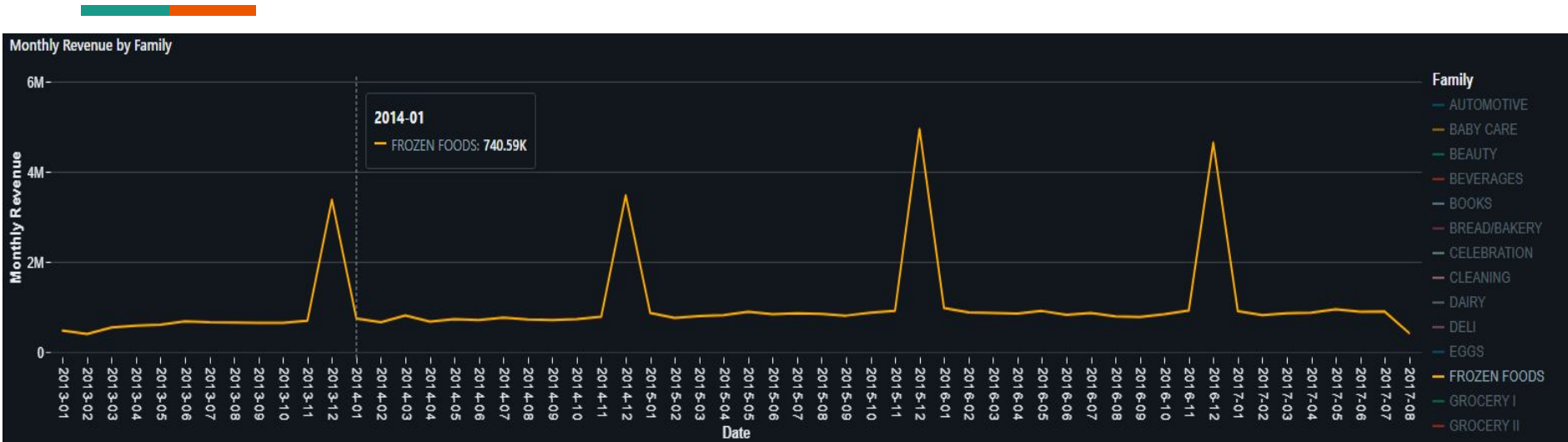
Promotions account for a major chunk of total sales.



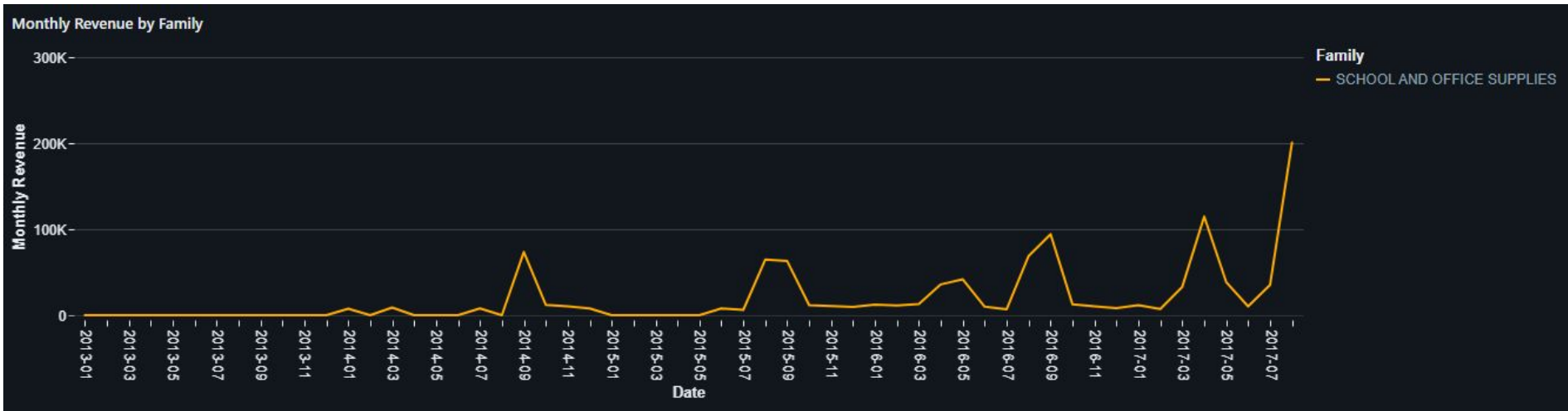
States like Azuay and Esmeraldas are severely impacted by holidays. Preventative measures should be taken



BI Dashboards - External Factors



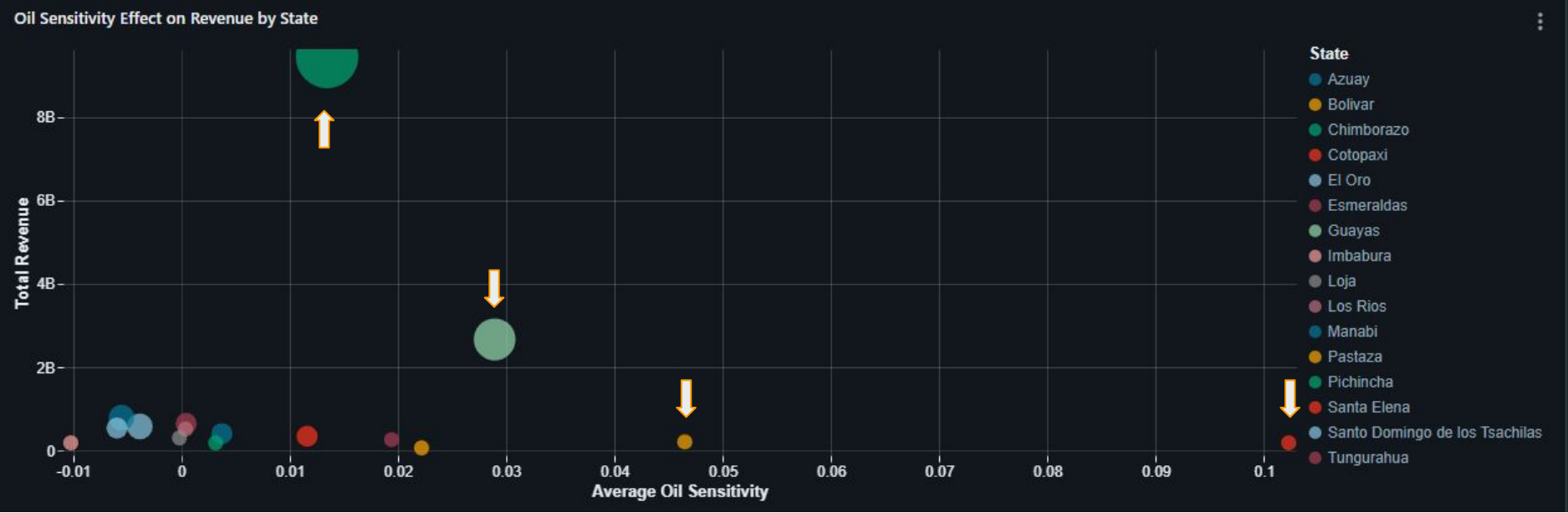
Frozen foods family of products sees a near 5x in sales every winter. Important to ensure inventory accounts for this surplus in sales



School supplies see an influx in purchases every August / September.



BI Dashboards - External Factors



Most states are not affected by oil sensitivity but four states are impacted to differing degrees





Insights and Business Recommendations

Insights

1. Product categories have seasonality
2. Promotions have an effect on sales
3. Holidays and oil prices have state-level impacts

Recommendations

1. Implement separate forecasting models for these area: seasonal category-level, state-level and perishable goods (include supply chain/inventory adjustments)
2. Coordinate promotions with forecasting models to prevent stockouts
3. Implement dynamic automated re-ordering systems with store-specific rules
4. Develop early-warning system for inventories trending to stockouts. Create contingency plans, cross-store inventory sharing and contract with backup suppliers for high-impact events





Planned Technology Refinements

1. Data:
 - a. Gather additional factors to broaden insights (weather, unemployment rate, GDP trend, demographics, marketing campaigns, store characteristics)
2. Technology:
 - a. Analyze workload and refine infrastructure needs, z-ordering, partitioning, cluster configuration, memory, executor/worker configuration
 - b. Monitor consumption trends to optimize costing and resources
 - c. Create additional dashboards targeting additional audience
 - d. Expand to external data sources (weather, government data)
3. People:
 - a. Work with team members to find friction points and improve collaboration, eg. working across disparate time zones





Q & A

Thank You





References

<https://www.kaggle.com/competitions/store-sales-time-series-forecasting/overview>

<https://www.leaplogic.io/blog/7-best-practices-to-modernize-data-architecture-on-databricks-with-leaplogic>

<https://www.databricks.com/blog/data-modeling-best-practices-implementation-modern-lakehouse>

<https://www.linkedin.com/pulse/migrating-from-traditional-databases-databricks-strategic-reddy-bgckc>





Appendix



Workflow

Workflows > Jobs >

CSCI E-103 Final Project Group 1 ☆

[Send feedback](#)

[Run now](#)

Runs **Tasks**

Data_Pipeline

...nal Project Group 1 - DE Data Pipeline

Shared Assignment Cluster

+ Add task

Task name*	<input type="text" value="Data_Pipeline"/>	
Type*	<div>Notebook</div>	
Source*	<div>Workspace</div>	
Path*	<div>...kspace/Shared/Final-Project-Group-1/Final Project Group 1 - DE Data Pipeline</div>	
Cluster*	<div>Shared Assignment Cluster</div> <div>DBR 14.3 LTS ML · Spark 3.5.0 · Scala 2.12</div>	

Job details

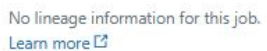
Job ID 501837888017184

Creator  Kwan, Richard

Run as  Kwan, Richard

Tags 

Description 

Lineage 

Git

Not configured



Schedules & Triggers

None



Compute

 Shared Assignment Cluster

Driver: m5d.large · Workers: m5d.large · 1-4 workers · On-demand and Spot · fall back to On-demand · DBR: 14.3 LTS ML (includes Apache Spark 3.5.0)





Model Registry

```
1 from mlflow.tracking import MlflowClient
2 import mlflow
3
4 def list_registered_models():
5     """List all registered models in the Model Registry"""
6     client = MlflowClient()
7
8     # Get all registered models
9     registered_models = client.search_registered_models()
10
11     for rm in registered_models:
12         print(f"Model: {rm.name}")
13         # Get latest versions for each stage
14         latest_versions = client.get_latest_versions(rm.name)
15         for version in latest_versions:
16             print(f"\tVersion: {version.version}")
17             print(f"\tStage: {version.current_stage}")
18             print(f"\tRun ID: {version.run_id}")
19             print(f"\tStatus: {version.status}")
20             print(f"\tDescription: {version.description}")
21
22 list_registered_models()
```

```
Model: richard_mackwan_fraud_detection_gbt
  Version: 1
  Stage: Production
  Run ID: af5daa739bae45dd8961b9b5c946160b
  Status: READY
  Description:
```

/root/.ipykernel/55594/command-1191953279993364-2913683411:14: FutureWarning: ``mlflow.tracking.client.MlflowClient.get_latest_versions`` is deprecated since 2.9.0. Model registry stages will be removed in a future major release. To learn more about the deprecation of model registry stages, see our migration guide here: <https://mlflow.org/docs/2.9.2/model-registry.html#migrating-from-stages>

```
latest_versions = client.get_latest_versions(rm.name)
```



Dashboard

