Democratizing ML feature store framework at scale

Rafay Aleem & Victoria Varney

Airflow Summit
Let's flow together

September 19-21, 2023,
Toronto, Canada

Rafay Aleem
Faire
Sr Data Engineer - ML platform

Victoria Varney
Astronomer
Sr Customer Success Manager

Today we are talking about....

Democratizing a machine learning **feature store framework** at Faire

How we have enabled **everyone** to **contribute** to a shared resource easily

Building a **framework** on top of Airflow by leveraging it's low-level **APIs**

How Airflow is **much more** than just a workflow management tool

F & Co S

FAIRE USA

F A I R E

FAIRE CO. FAIRE CO. *f*

F A I R E

san francisco

CA F

f

FA IRE

F A I R E

FA IRE

FAIRE CO. FAIRE CO. *the*

F

faire

san francisco

f

the future

FA IR E

is local

FAIRE CO. FAIRE CO.

f

— *f a i r e* —

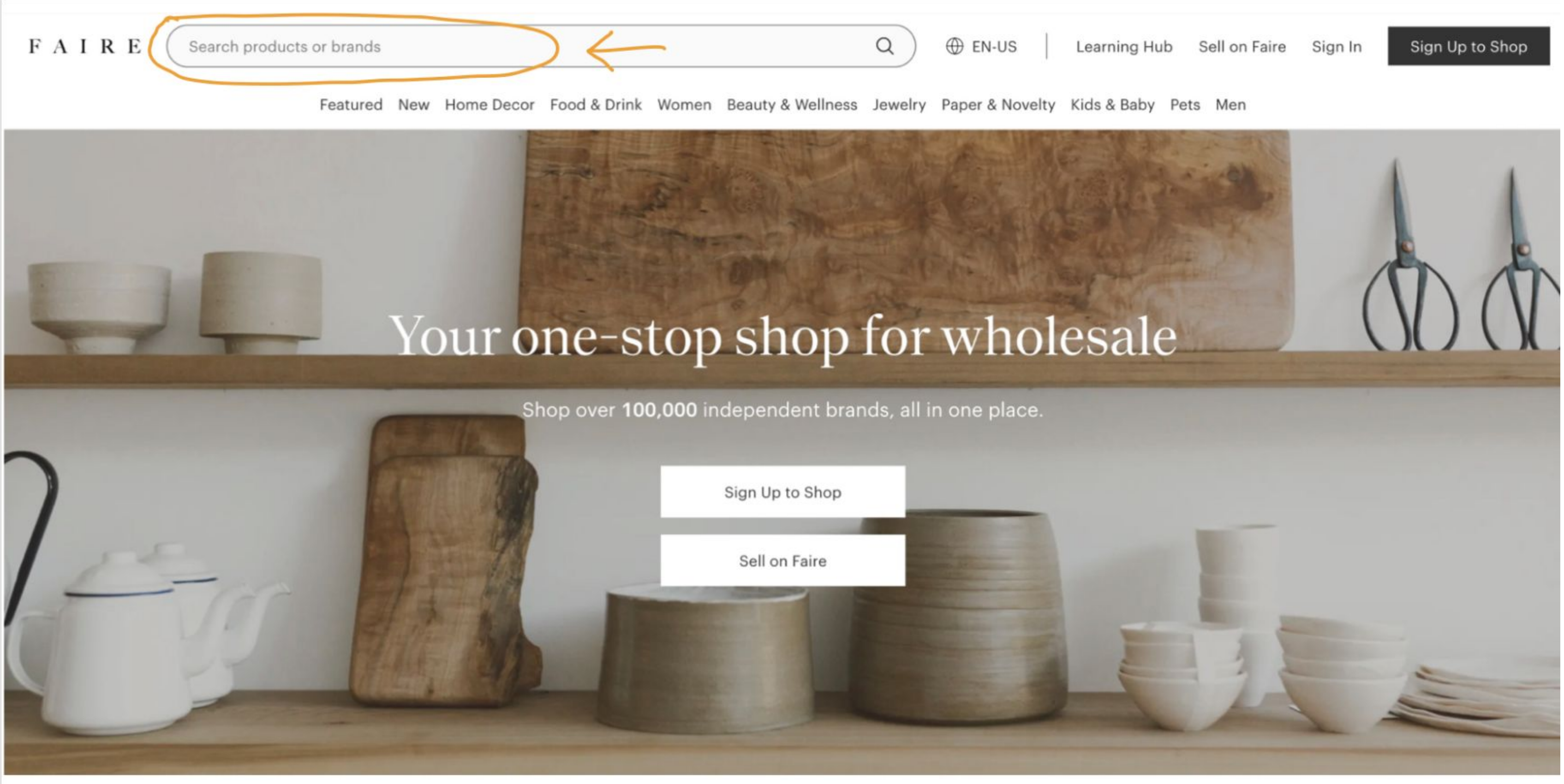
F A I R E

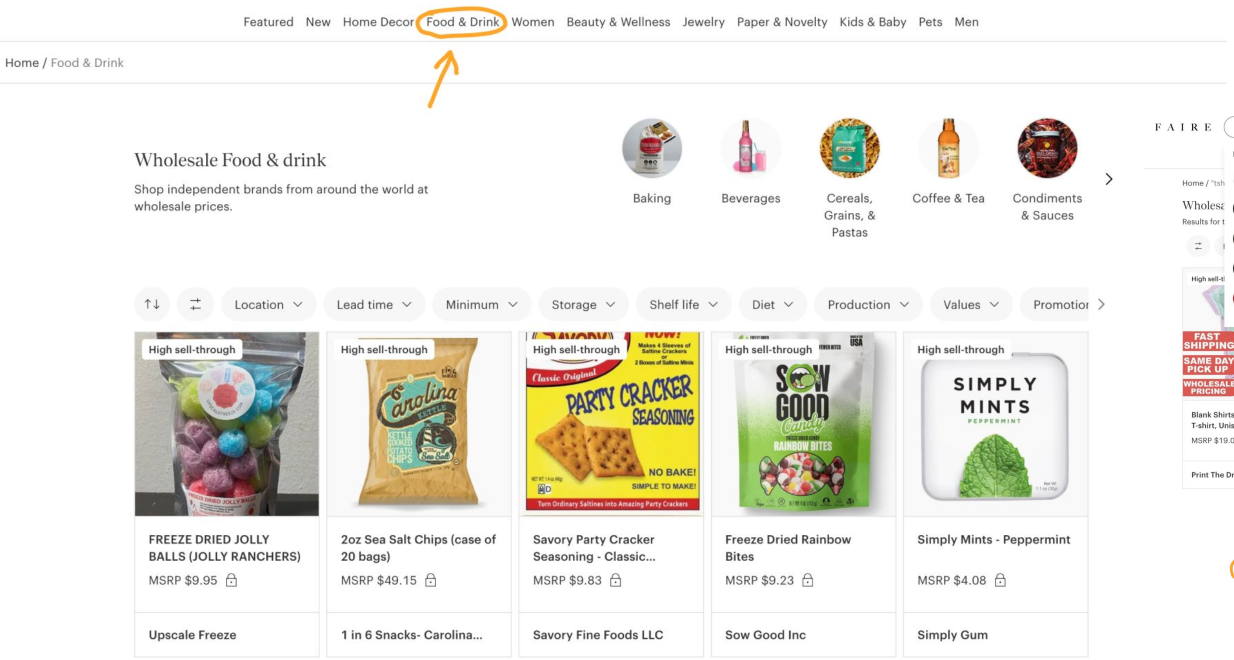
san francisco

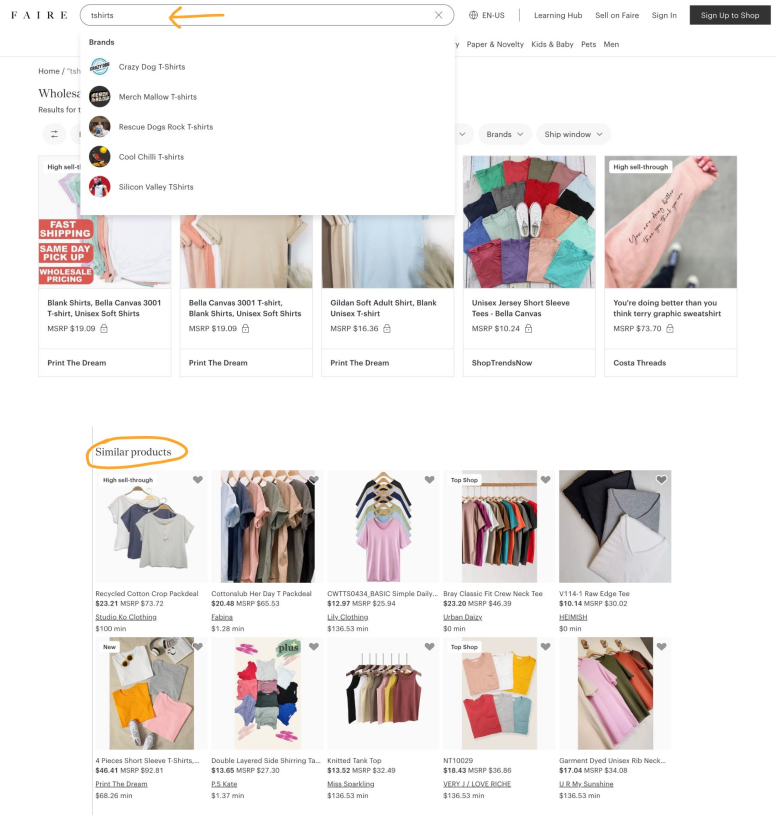
f a i r e

san francisco

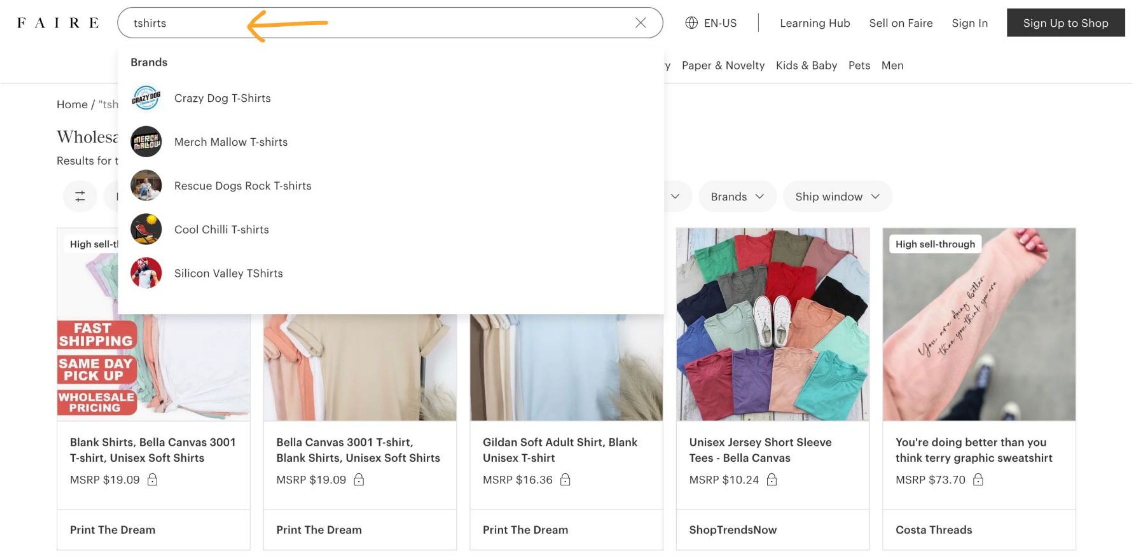
Search experience at Faire

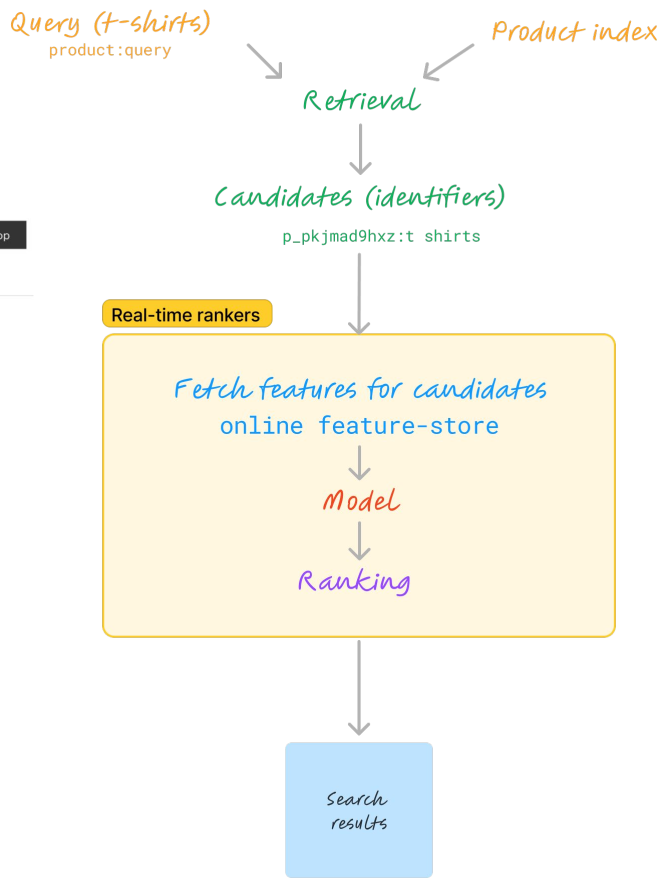



Featured New Home Decor Food & Drink Women Beauty & Wellness Jewelry Paper & Novelty Kids & Baby Pets Men
Home / Food & Drink
Wholesale Food & drink
Shop independent brands from around the world at wholesale prices.
Baking
Beverages
Cereals, Grains, & Pastas
Coffee & Tea
Condiments & Sauces
Location
Lead time
Minimum
Storage
Shelf life
Diet
Production
Values
Promotior
High sell-through
FREEZE DRIED JOLLY BALLS (JOLLY RANCHERS)
MSRP $9.95
Upscale Freeze
High sell-through
2oz Sea Salt Chips (case of 20 bags)
MSRP $49.15
1 in 6 Snacks- Carolina...
High sell-through
Savory Party Cracker Seasoning - Classic...
MSRP $9.83
Savory Fine Foods LLC
High sell-through
Freeze Dried Rainbow Bites
MSRP $9.23
Sow Good Inc
High sell-through
Simply Mints - Peppermint
MSRP $4.08
Simply Gum

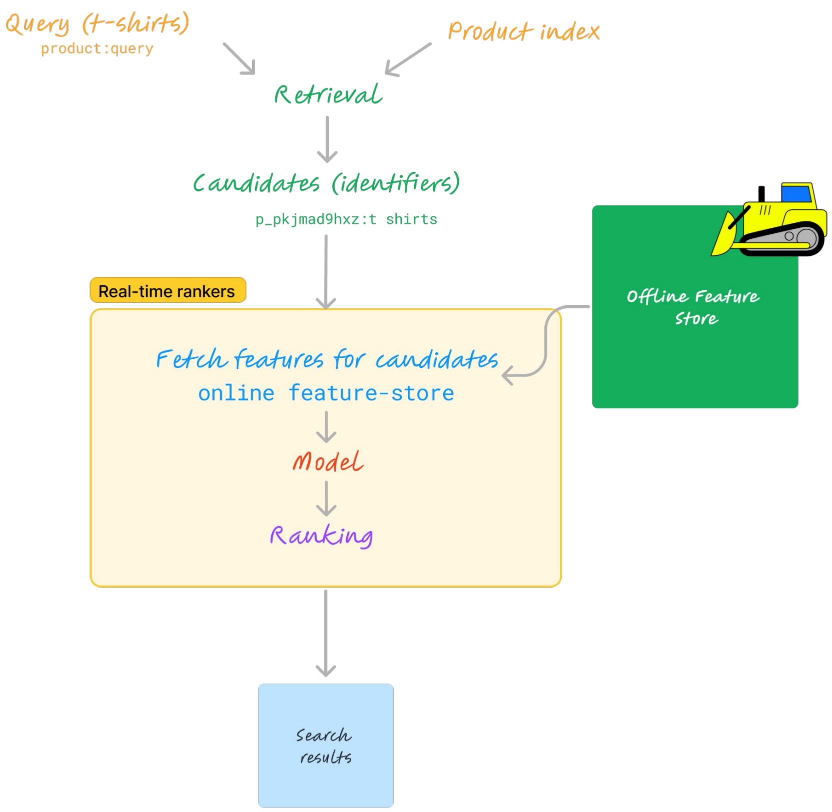

FAIRE
tshirts
EN-US
Learning Hub
Sell on Faire
Sign In
Sign Up to Shop
Brands
Crazy Dog T-Shirts
Merch Mallow T-shirts
Rescue Dogs Rock T-shirts
Cool Chilli T-shirts
Silicon Valley TShirts
Paper & Novelty Kids & Baby Pets Men
Brands
Ship window
FAST SHIPPING
SAME DAY PICK UP
WHOLESALE PRICING
Blank Shirts, Bella Canvas 3001 T-shirt, Unisex Soft Shirts
MSRP $19.09
Print The Dream
Bella Canvas 3001 T-shirt, Blank Shirts, Unisex Soft Shirts
MSRP $19.09
Print The Dream
Gildan Soft Adult Shirt, Blank Unisex T-shirt
MSRP $16.36
Print The Dream
Unisex Jersey Short Sleeve Tees - Bella Canvas
MSRP $10.24
ShopTrendsNow
High sell-through
You're doing better than you think terry graphic sweatshirt
MSRP $73.70
Costa Threads
Similar products
High sell-through
Recycled Cotton Crop Packdeal
$23.21 MSRP $73.72
Studio Ko Clothing
$100 min
Cottonslub Her Day T Packdeal
$20.48 MSRP $65.53
Fabina
$1.28 min
CWTTS0434_BASIC Simple Daily...
$12.97 MSRP $25.94
Lily Clothing
$136.53 min
Top Shop
Bray Classic Fit Crew Neck Tee
$23.20 MSRP $46.39
Urban Daizy
$0 min
V114-1 Raw Edge Tee
$10.14 MSRP $30.02
HEIMISH
$0 min
New
4 Pieces Short Sleeve T-Shirts,...
$46.41 MSRP $92.81
Print The Dream
$68.26 min
Double Layered Side Shirring Ta...
$13.65 MSRP $27.30
P.S Kate
$1.37 min
Knitted Tank Top
$13.52 MSRP $32.49
Miss Sparkling
$136.53 min
Top Shop
NT10029
$18.43 MSRP $36.86
VERY J / LOVE RICHE
$136.53 min
Garment Dyed Unisex Rib Neck...
$17.04 MSRP $34.08
U R My Sunshine
$136.53 min

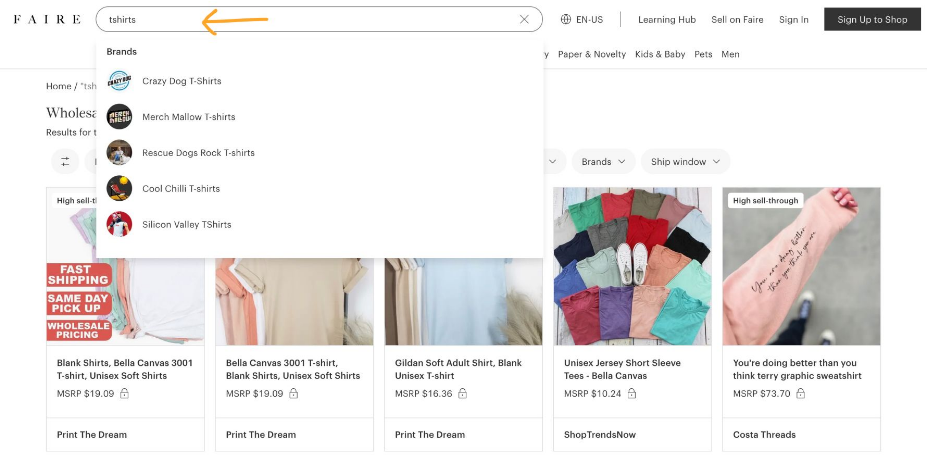
Anatomy of a search





Anatomy of a search

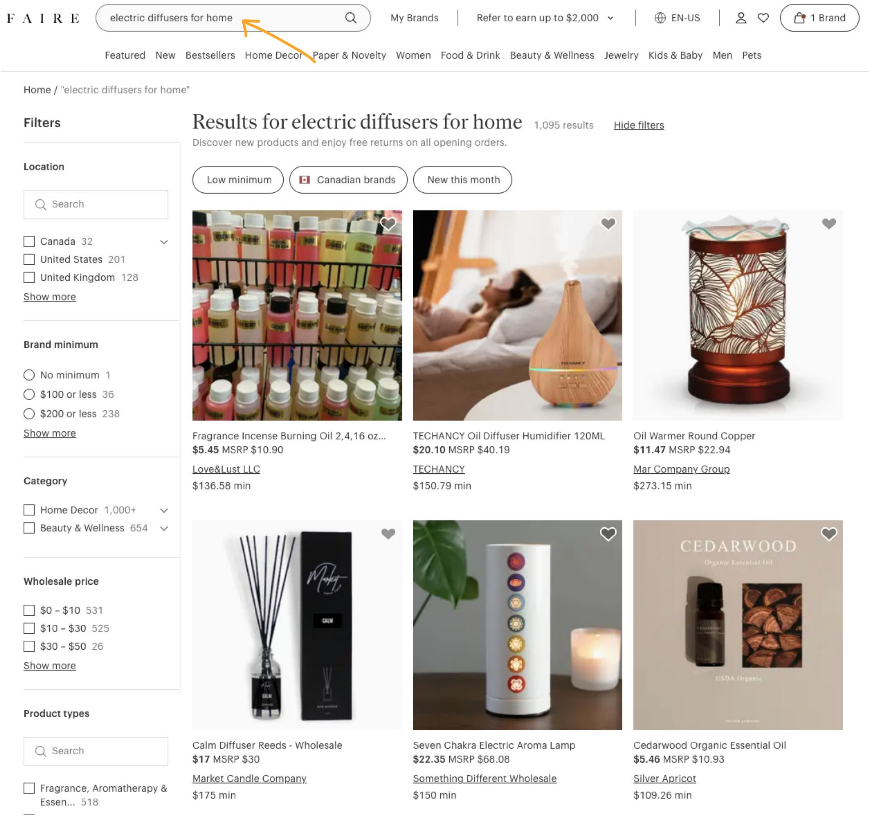




Enhancing the search experience

- **Unlock** the science behind **search relevancy**

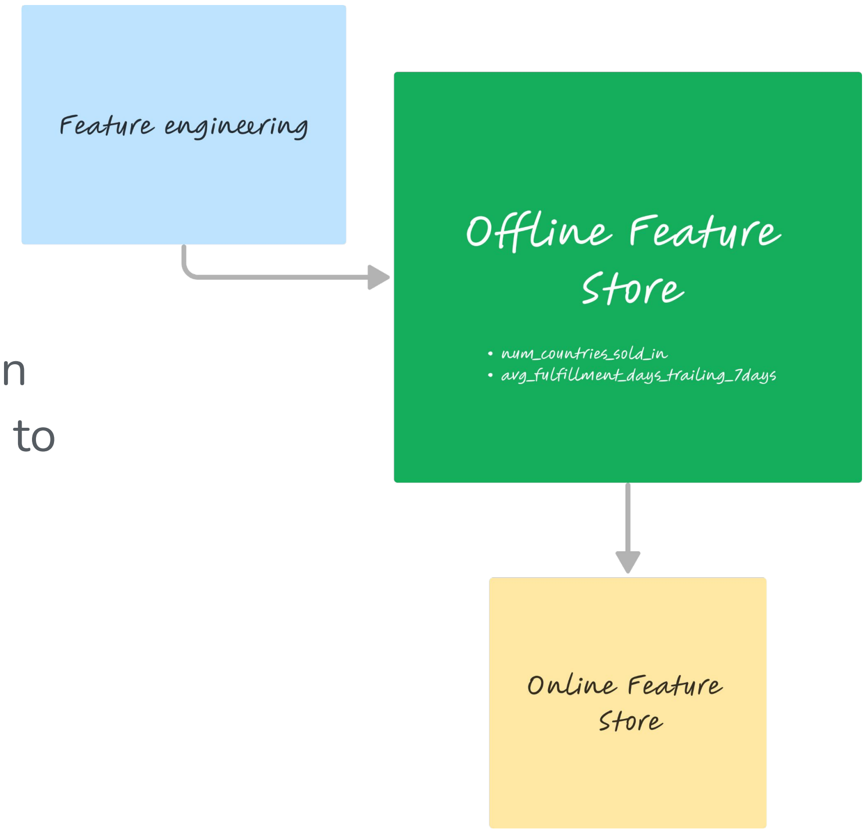
- Search is **particularly important** in context of e-Commerce



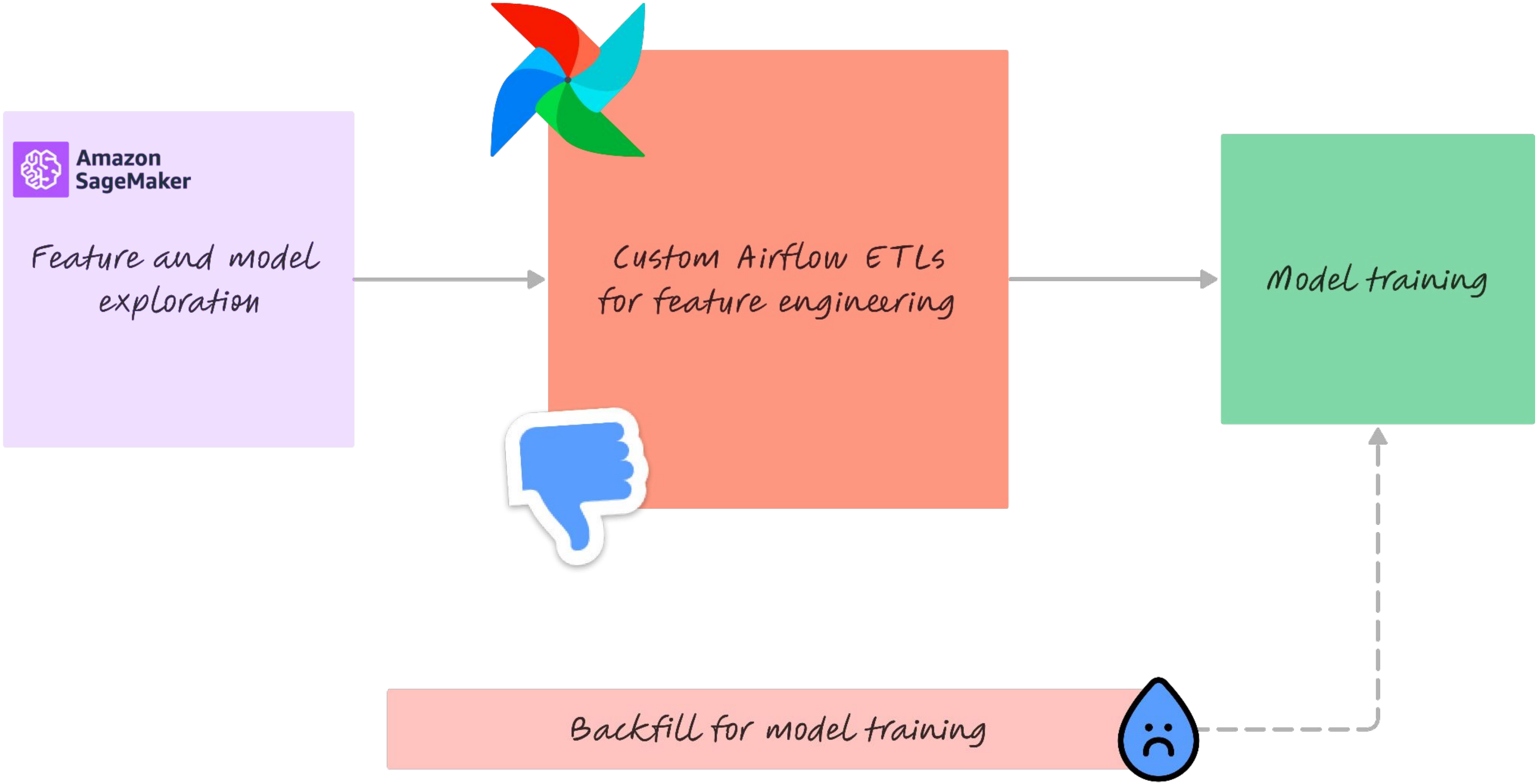
Enhancing the search experience

- Features are **float values** with some **underlying meaning** such as
  - num_countries_sold_in
  - avg_fulfillment_days_trailing_7_days

- Features are engineered and defined on the **offline** and eventually **propagated** to **online** feature store

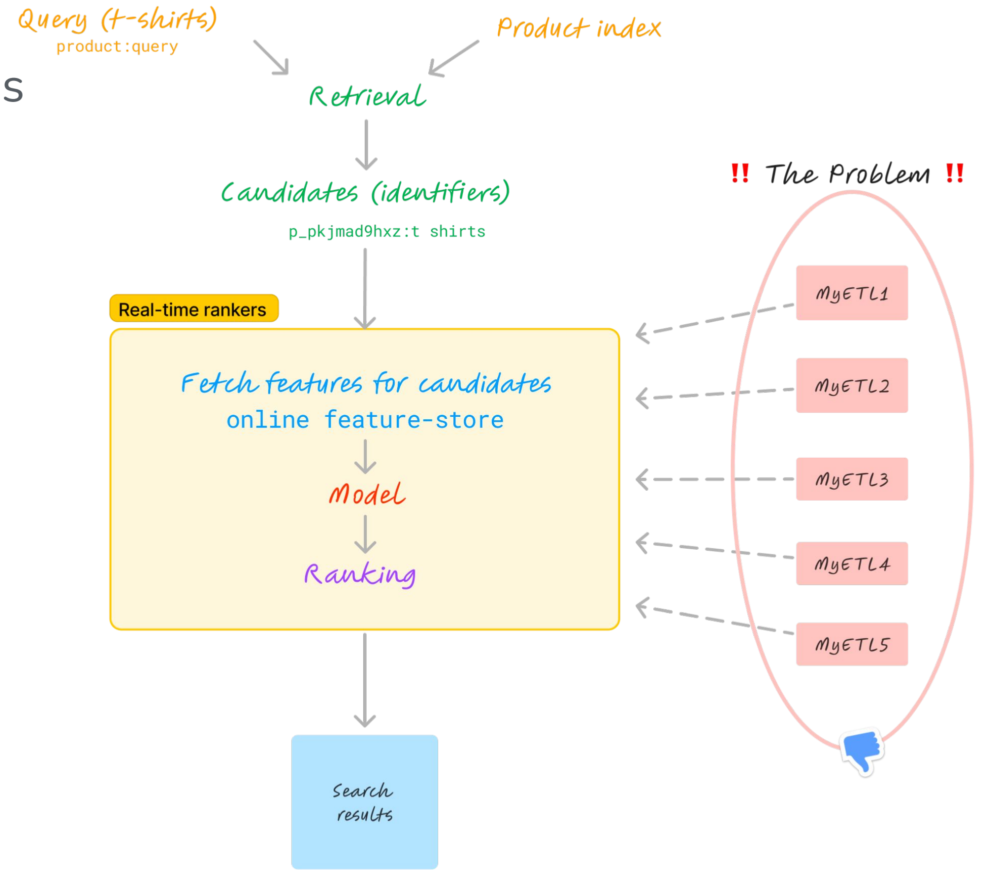


The need for democratizing the feature store

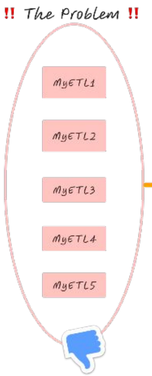


The need for democratizing the feature store

- No **feature visibility** across teams
- No clear process to separate **online features** from offline
- Non-standardized **error-prone backfilling** with no clear notion of point-in-time joins
- Overhead **costs** on ElasticCache and Snowflake



The need for democratizing the feature store



Consolidate all feature definitions into a single feature store with a complete feature registry

Complex
&
Error-prone
backfills

Simplify and standardize feature backfilling

SQL
+
Custom ETLs

```
SELECT
    SUM(AMOUNT) AS TOTAL_SALES
FROM ORDERS
    WHERE CATEGORY = “candles”
```

Plain SQL as feature definitions

Feature Store Framework

- All features are defined as **SQL** with their **Python configurations**

- Each feature can be **configured separately** with its own **metadata**

  Examples
  - specify if available online
  - feature description
  - author

- All configuration is fed into a queryable **feature registry**

```sql
select
    b.token as brand_token,
    count(distinct rb.retailer_id) as number_of_contact_books_found
from production.retailer_to_brand_email_domains rb
join production.brands b
    on b.url_domain = rb.email_domain
where b.active
    and rb.created_at < '{{ ds }}'::DATE
group by 1
```

```python
feature_groups = [
    FeatureTask(
        sql_file="brand_contact_book.sql",
        features=[
            Feature(
                sql_feature_name="number_of_contact_books_found",
                feature_description="Number of retailers with this brand in their address book",
                available_online=False,
            )
        ],
        entity=BrandEntity(),
        author="rafay",
        is_static=False,
    )
]

brand_features = FeatureEntity(entity=BrandEntity(), tasks=feature_groups)
```

Feature Store Framework

```python
class FeatureEntity:
    def __init__(
        self,
        entity: EntityDescription,
        tasks: list[FeatureTask],
    ):
        self.entity = entity
        self.tasks = tasks
        self.name = entity.get_feature_name_string()

        for task in tasks:
            assert type(self.entity) == type(task.entity)
            self.check_columns_present_in_sql(task)

        # check dup feature names
        self.check_dup_feature_names()

    def check_dup_feature_names(self) -> None:
        """
        Check whether feature names are unique.
        """
        ...

    def get_compute_operator(
        self, task: FeatureTask, is_backfill=False, start_date=None, end_date=None
    ) -> MLFeatureSnowflakeOperator:
        ...
```

Low-level APIs

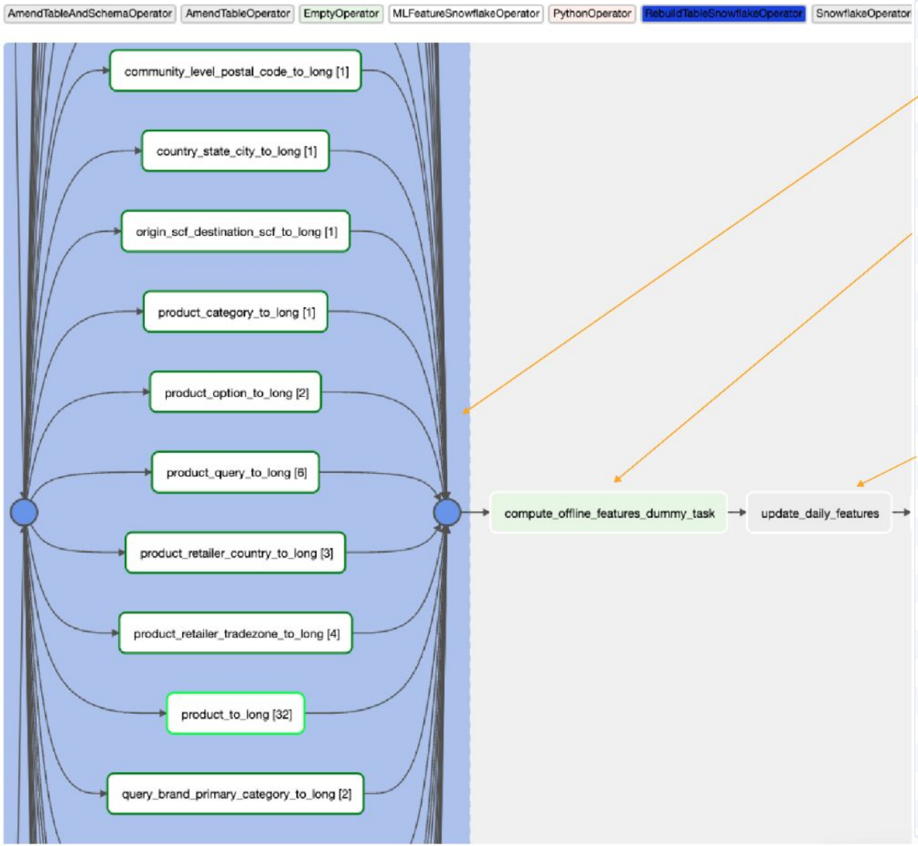
```python
@dataclass
class Feature:
    sql_feature_name: str
    feature_description: str = "No description has been specified"
    available_online: bool = False
```

```python
@dataclass
class FeatureTask:
    sql_file: str
    features: list[Feature]
    entity: EntityDescription
    author: str = "Unknown author"
    is_static: bool = False

    # backfill parameters
    backfill_config: FeatureTaskBackfillConfig = None
```

High-level config dataclasses

Feature Store Framework




```python
# This step creates individual tables for each invocation of MLFeatureSnowflakeOperator
# You can find these tables in {schema_name}.features_{feature_group_name}
with TaskGroup(group_id="compute_offline_features") as compute_offline_features:
    for feature_entity in FEATURE_ENTITIES:
        feature_entity.get_dynamically_mapped_tasks(config)

# This is a dummy task used to make it easier to manage the DAG in the Airflow UI
compute_offline_features_dummy_task = EmptyOperator(
    task_id="compute_offline_features_dummy_task",
    doc="Dummy task to trigger offline features computation",
    trigger_rule="all_done",
)

# This pulls the daily feature used for dispersal and pivoting.
# Data is persisted to avoid duplicated computation and facilitate data checking.
update_daily_features = AmendTableOperator(
    task_id="update_daily_features",
    schema_name=config.get("schema"),
    table_name=config.get("features_daily_table_name"),
    create_sql="sql/create_features_daily_table.sql",
    sql="sql/update_features_daily_table.sql",
    fill_condition="ds = '{{ ds }}'",
    data_checks=[
        f"checks/{check}.sql"
        for check in (
            "update_daily_features_task_lands_data",
        )
    ]
)
```

Adopting new Airflow features

Merged **ML-314: Fix slow dag parsing in compute offline feature store dag that is causing zombie tasks** #17037

Changes from all commits · File filter · Conversations · 0 / 3 files viewed · Review changes

48 airflow/dags/feature_store/features/utils/FeatureEntity.py · Viewed

```diff
-    def get_compute_operators(self) -> list[MLFeatureSnowflakeOperator]:
         """
-        Generate list of Airflow tasks to perform the computation of all feature tasks
         """
-        return [self.get_compute_operator(task) for task in self.tasks]

-    def get_compute_task_group(self) -> TaskGroup:
-        with TaskGroup(group_id=self.name) as task_group:
-            self.get_compute_operators()
-        return task_group
```

```diff
+    def get_dynamically_mapped_tasks(self, params) -> MappedOperator:
         """
+        Generate list of dynamically mapped Airflow tasks to perform the computation of all feature tasks
+
+        Args:
+            params: This is the same config that's passed to the DAG params. The reason we have to explicitly
+                pass it here is because expand_kwargs would not render the templated fields in the operator
+                when using it like {{ params.features_static_table_name }} in combination with partial and
+                expand_kwargs. Also note that expand_kwargs expects a list of dictionaries with string
+                key value pairs.
         """

+        # Please see https://docs.astronomer.io/learn/dynamic-tasks?tab=traditional#sets-of-keyword-arguments
+        # for understanding partial and expand_kwargs in Airflow's Dynamic Task Mapping API
+        #
+        # expand_kwargs takes the following form when using Dynamic Task Mapping API
+        # [
+        #     {"bash_command": "echo $WORD", "env": {"WORD": "hello"}},
+        #     {"bash_command": "echo `expr length $WORD`", "env": {"WORD": "tea"}},
+        #     {"bash_command": "echo ${WORD//e/X}", "env": {"WORD": "goodbye"}},
+        # ]
+
+        kwargs = list(
+            map(
+                lambda t: {
+                    "feature_task": t,
+                    "destination_table_name": params["features_static_table_name"] if t.is_static
```

Status: Merged

Eason6wang marked this conversation as resolved. · Show resolved

```diff
+                    else params["features_long_table_name"]
+                },
+                self.tasks
+            )
+        )
+
+        # We use partial and expand_kwargs here to optimize dag parsing and delay task generation to runtime
+        return MLFeatureSnowflakeOperator.partial(
+            task_id=f"{self.name}_to_long",
+            start_date=None,
+            end_date=None,
```

```python
def get_dynamically_mapped_tasks(self, params) -> MappedOperator:
    kwargs = list(
        map(
            lambda t: {
                "feature_task": t,
                "destination_table_name": params["features_static_table_name"] if t.is_static
                else params["features_long_table_name"]
            },
            self.tasks
        )
    )

    # We use partial and expand_kwargs here to optimize dag parsing and delay task generation to runtime
    return MLFeatureSnowflakeOperator.partial(
        task_id=f"{self.name}_to_long",
        start_date=None,
        end_date=None,
        is_backfill=False,
    ).expand_kwargs(kwargs)
```

Use partial *and* expand_kwargs *to dynamically map existing task*

Adopting new Airflow features

- Be **proactive** in adopting new versions of Airflow
- New features have greatly **improved** cluster performance
- Notable features:
    - **Dynamic Task Mapping**
    - **Task Groups** (easier visual dag management)
    - **Deferrable operators** for long running tasks such as AWS Batch jobs

Extending the framework for feature backfills

```python
FeatureTask(
    sql_file="brand_contact_book.sql",
    features=[
        Feature(
            sql_feature_name="number_of_contact_books_found",
            feature_description="Number of retailers with this brand in their address book",
            available_online=False,
            backfill_config=FeatureTaskBackfillConfig(
                start_date=datetime(2023, 8, 1),
                end_date=datetime(2023, 8, 30),
                namespace="brand_1_month_backfill",
            ),
        )
    ],
    entity=BrandEntity(),
    author="rafay",
    is_static=False,
)
```

```
feature_store_backfill_brand_1_month_2023_8_1_2023_8_30
```

- Uses same SQL files
- Provides extra jinja templated flag {{ is_backfill }}
- Entire process takes 3 lines of configuration!

Internally passed to low-level API

```python
def get_compute_operator(
    self, task: FeatureTask, is_backfill=False, start_date=None, end_date=None
) -> MLFeatureSnowflakeOperator:
    """
    Generate Airflow task to perform the computation of a feature task

    Args:
        task: FeatureTask object
        is_backfill: specifies whether the operator should be generated for backfill
        start_date: start date of the task, None implies it will match the dag start date
        end_date: end date of the task, None implies it will match the dag end date
    """
    destination_table_name = (
        "{{ params.features_static_table_name }}"
        if task.is_static
        else "{{ params.features_long_table_name }}"
    )
    return MLFeatureSnowflakeOperator(
        task_id=task.sql_name,
        feature_task=task,
        destination_table_name=destination_table_name,
        start_date=start_date,
        end_date=end_date,
        is_backfill=is_backfill,
    )  # type: ignore
```

Best practices for designing extensible frameworks

- Use mixins to extend operator capabilities

```
class RebuildTableSnowflakeOperator(
    SnapshotMixin, TableOperatorMixin, ETLSnowflakeOperator
):
```

Extend operator capabilities using mixins

- Allows adding new features to low-level APIs without breaking

```
class SnapshotConfig:
    Frequency = Enum('Frequency', ['NEVER', 'DAILY', 'EVERY_RUN'])
```

```
store_daily_embeddings = RebuildTableSnowflakeOperator(
    task_id="store_daily_embeddings",
    sql=daily_online_embedding_sql(),
    schema_name=config.get("schema"),
    table_name="daily_embeddings_v2",
    snapshot_frequency=SnapshotConfig.Frequency.EVERY_RUN,
)
```

Enable table snapshots using predefined frequency

Airflow beyond workflow management

Airflow goes **far and beyond** a workflow management tool

It's **thoughtful design** makes it very extensible and powerful

It is very well suited for running **mission critical** workflows with tight SLA requirements

It continues to be a very stable part of **Faire's infrastructure** and continue to scale

Key Takeaways

Proactively consider onboarding to **latest Airflow** versions

Don't think of Airflow as just a **workflow management** tool

Consider building **shared frameworks** instead of shared ETLs

Thoughtful use of Airflow **APIs** and **features** goes a long way

Credits

- Wayne Zhang for his guidance on the offline feature store framework
- **Analytics Engineering** team for their feedback on table snapshot tooling
- **Core Data Infra** team for their constant support with Airflow and Snowflake
- **Machine Learning Platform** team for dealing with on-call issues and providing stakeholder support
- My wife's constant support

Wrap Up & Questions

- Careers @ Faire: faire.com/careers/

- Where to find Rafay
  - LinkedIn: linkedin.com/in/mrafayaleem/
  - Airflow Slack: @Rafay Aleem
  - Twitter: @mrafayaleem
  - Email: contact@mrafayaleem.com
  - Newsletter: mrafayaleem.com/#/portal/

Thank You!