Fennel

Realtime Feature Engineering Platform

Nikhil Garg, CEO

nikhil@fennel.ai

Aditya Nambiar
Cofounder, Ex-Facebook, Google

Nikhil Garg
CEO, Ex-Facebook, Quora

Abhay Bothra
CTO, Ex-Facebook, Libra

Neha Narkhede
Creator of Kafka,
Co-founder Confluent

John Hegeman
SVP, Facebook Ads

Adam D'Angelo
Ex-CTO Facebook,
OpenAI Board, CEO Quora

Kevin Weil
CPO OpenAI, VP Twitter

Mikhail Parakhin
Ex-CTO Yandex,
CEO Microsoft Bing

Xavier Amatriain
VP Google. LinkedIn,

1. Ease of Authoring Features

- DS teams depend on engineering
- Need Python/Pandas native tooling
- Two definitions for offline/online, backfilling

2. Realtime ML

- Freshness of features
- Low latency serving

3. Data/Feature Quality

- Preventive: Unit testing, CI/CD etc.
- Diagnostic: Drift, data expectations

4. Feature Reuse

- Standardization of tooling
- Discovery, catalog, health checks

5. Compliance

- Data privacy, infosec
- PII / RBAC, GDPR etc.

6. Cost

- Operational burden
- Cloud infra costs

Problems with Feature Engineering

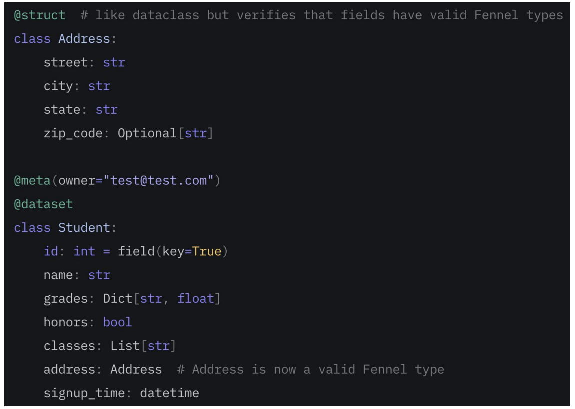
Data/Feature Quality: Why?

- At scale, some thing or the other is going wrong all the time.
- Result: models don’t perform as well, hard to debug regressions
- Usually an afterthought for in-house systems

Fennel’s Pillars of Data/Feature Quality

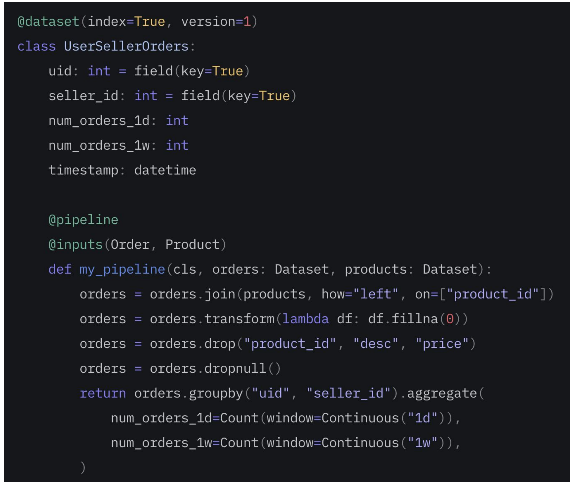
1. Strong Typing
2. Versioning, immutability
3. Unit Testing
4. Compile Time Lineage Validation
5. Structured metadata & ownership
6. Branches
7. Data expectations
8. Feature drift detection

1. Strong Typing



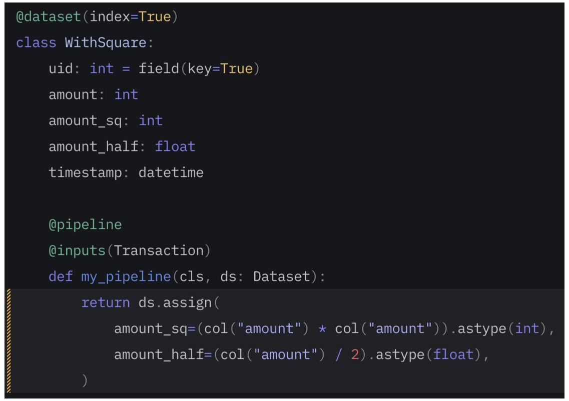
```python
@struct  # like dataclass but verifies that fields have valid Fennel types
class Address:
    street: str
    city: str
    state: str
    zip_code: Optional[str]

@meta(owner="test@test.com")
@dataset
class Student:
    id: int = field(key=True)
    name: str
    grades: Dict[str, float]
    honors: bool
    classes: List[str]
    address: Address  # Address is now a valid Fennel type
    signup_time: datetime
```

Rich type system



```python
@dataset(index=True, version=1)
class UserSellerOrders:
    uid: int = field(key=True)
    seller_id: int = field(key=True)
    num_orders_1d: int
    num_orders_1w: int
    timestamp: datetime

    @pipeline
    @inputs(Order, Product)
    def my_pipeline(cls, orders: Dataset, products: Dataset):
        orders = orders.join(products, how="left", on=["product_id"])
        orders = orders.transform(lambda df: df.fillna(0))
        orders = orders.drop("product_id", "desc", "price")
        orders = orders.dropnull()
        return orders.groupby("uid", "seller_id").aggregate(
            num_orders_1d=Count(window=Continuous("1d")),
            num_orders_1w=Count(window=Continuous("1w")),
        )
```

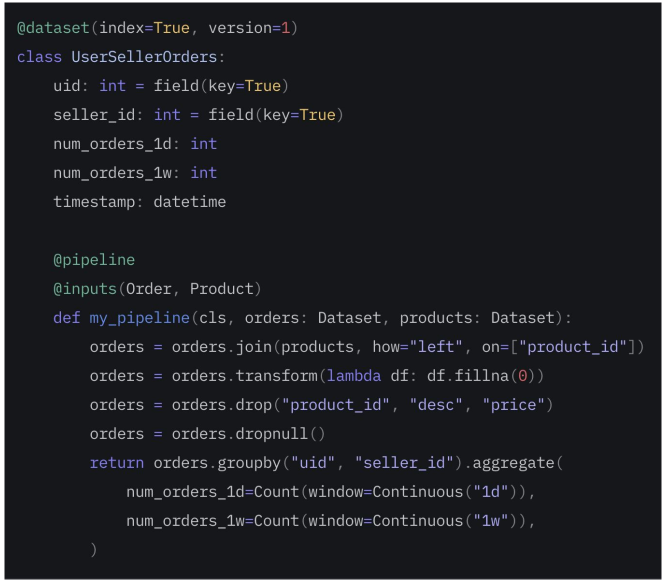
Strongly typed pipelines



```python
@dataset(index=True)
class WithSquare:
    uid: int = field(key=True)
    amount: int
    amount_sq: int
    amount_half: float
    timestamp: datetime

    @pipeline
    @inputs(Transaction)
    def my_pipeline(cls, ds: Dataset):
        return ds.assign(
            amount_sq=(col("amount") * col("amount")).astype(int),
            amount_half=(col("amount") / 2).astype(float),
        )
```

Strongly typed expressions

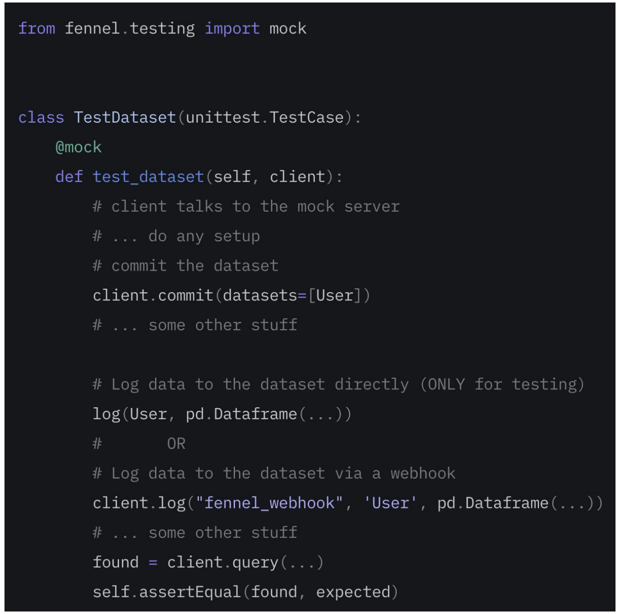
2. Versioning & Immutability



```python
@dataset(index=True, version=1)
class UserSellerOrders:
    uid: int = field(key=True)
    seller_id: int = field(key=True)
    num_orders_1d: int
    num_orders_1w: int
    timestamp: datetime

    @pipeline
    @inputs(Order, Product)
    def my_pipeline(cls, orders: Dataset, products: Dataset):
        orders = orders.join(products, how="left", on=["product_id"])
        orders = orders.transform(lambda df: df.fillna(0))
        orders = orders.drop("product_id", "desc", "price")
        orders = orders.dropnull()
        return orders.groupby("uid", "seller_id").aggregate(
            num_orders_1d=Count(window=Continuous("1d")),
            num_orders_1w=Count(window=Continuous("1w")),
        )
```

Assets are versioned and immutable

3. Unit Testing

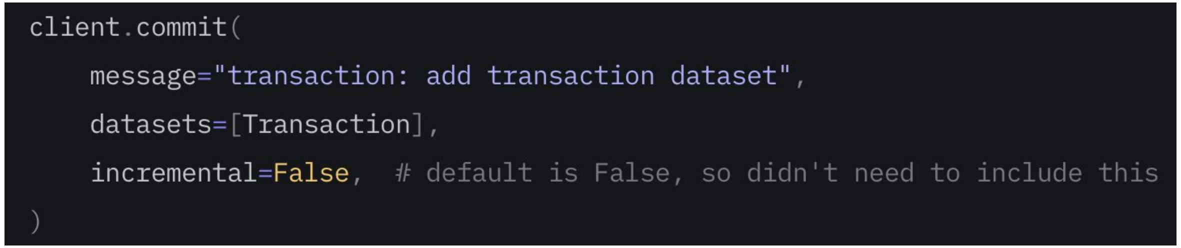


```
from fennel.testing import mock


class TestDataset(unittest.TestCase):
    @mock
    def test_dataset(self, client):
        # client talks to the mock server
        # ... do any setup
        # commit the dataset
        client.commit(datasets=[User])
        # ... some other stuff

        # Log data to the dataset directly (ONLY for testing)
        log(User, pd.Dataframe(...))
        #       OR
        # Log data to the dataset via a webhook
        client.log("fennel_webhook", 'User', pd.Dataframe(...))
        # ... some other stuff
        found = client.query(...)
        self.assertEqual(found, expected)
```

Entirety of Fennel is unit-testable

4. Compile Time Lineage Validations

1. Deleting something that others depend on
2. Changing something without updating the version
3. Type mismatches
4. Circular dependency
5. ..lot more



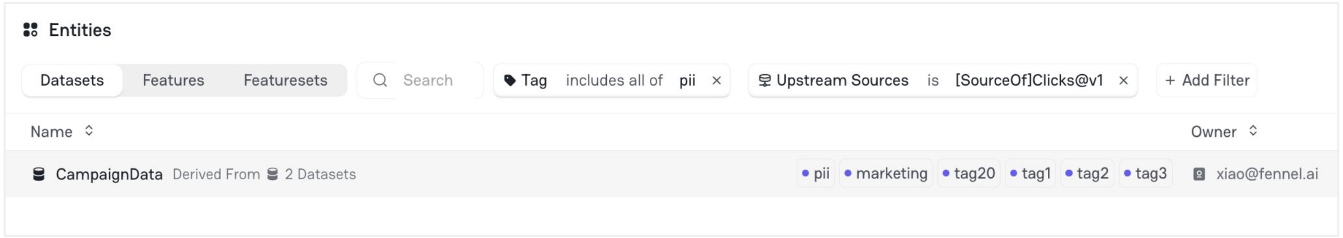
```python
client.commit(
    message="transaction: add transaction dataset",
    datasets=[Transaction],
    incremental=False,  # default is False, so didn't need to include this
)
```

Explicit commit operation that checks full lineage validity

5. Structured Metadata & Ownership

```
@meta(owner='feed-team@xyz.ai')
@featureset
class UserFeatures:
    uid: int = F()
    zip: str = F().meta(tags=['PII'])
    bmi: float = F().meta(owner='alan@xyz.ai')
    bmr: float = F().meta(deprecated=True)
```

Explicit ownership, tags, lifecycle status



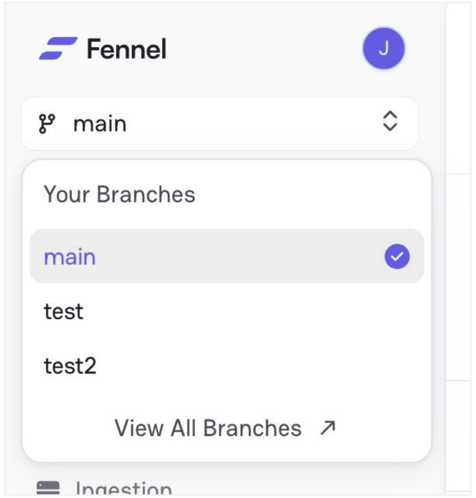
Entities

Datasets | Features | Featuresets | Search | Tag includes all of pii × | Upstream Sources is [SourceOf]Clicks@v1 × | + Add Filter

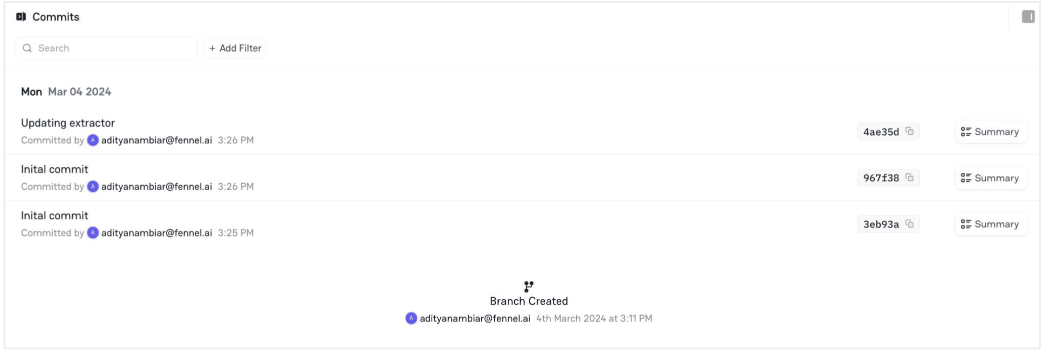
Name		Owner
CampaignData Derived From 2 Datasets	pii, marketing, tag20, tag1, tag2, tag3	xiao@fennel.ai

Rich discovery using metadata

6. Branches



Browsing all visible branches


Commits

Search

+ Add Filter

Mon Mar 04 2024

Updating extractor
Committed by adityanambiar@fennel.ai 3:26 PM — 4ae35d — Summary

Inital commit
Committed by adityanambiar@fennel.ai 3:26 PM — 967f38 — Summary

Inital commit
Committed by adityanambiar@fennel.ai 3:25 PM — 3eb93a — Summary

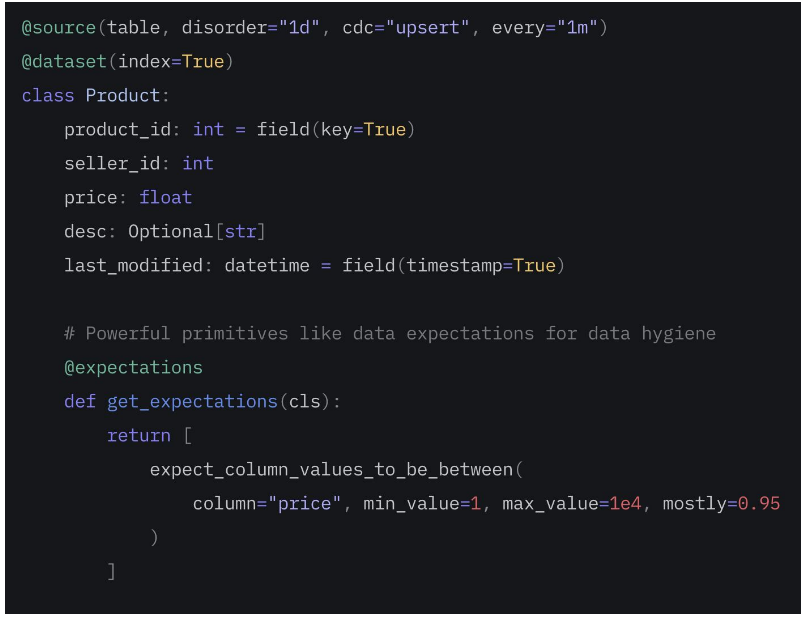
Branch Created
adityanambiar@fennel.ai 4th March 2024 at 3:11 PM

Full commit history of every branch

```
client.init_branch("dev")
client.commit(
    message="some module: some git like commit message",
    datasets=[SomeDataset],
    featuresets=[SomeFeatureset],
)
```

Code based changes to branches

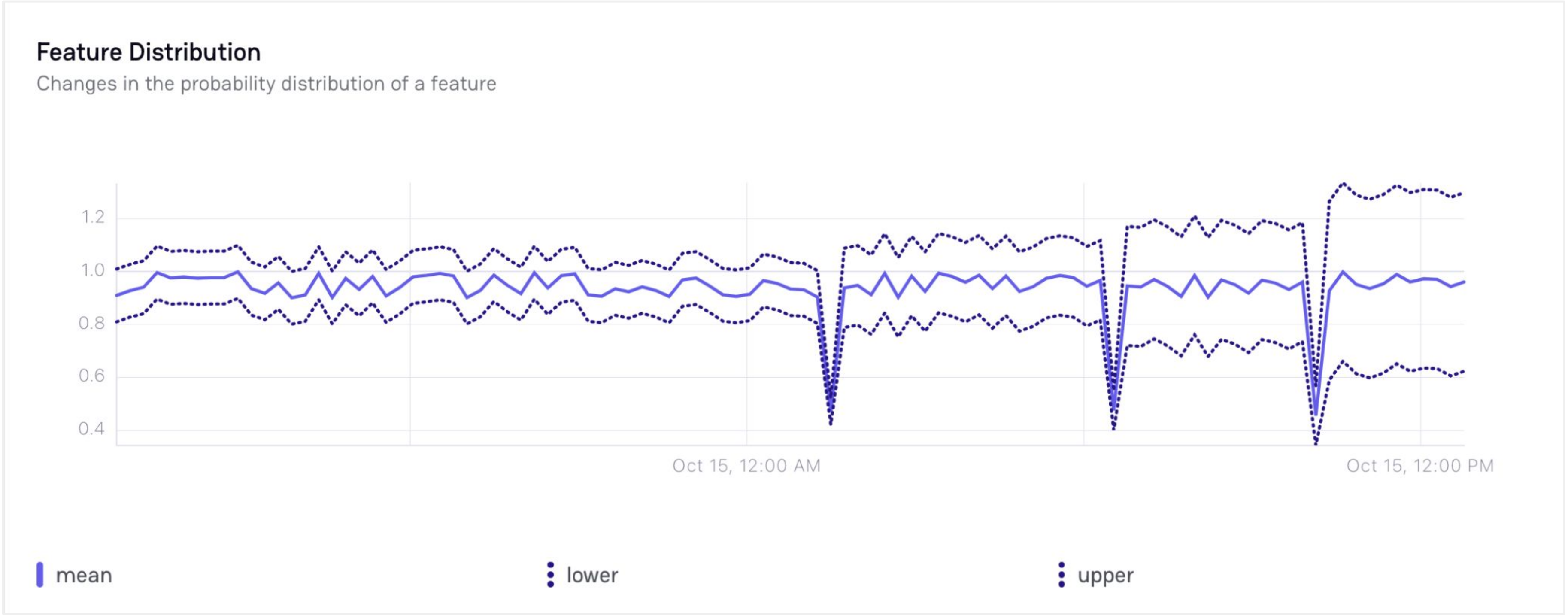
7. Data Expectations



```python
@source(table, disorder="1d", cdc="upsert", every="1m")
@dataset(index=True)
class Product:
    product_id: int = field(key=True)
    seller_id: int
    price: float
    desc: Optional[str]
    last_modified: datetime = field(timestamp=True)

    # Powerful primitives like data expectations for data hygiene
    @expectations
    def get_expectations(cls):
        return [
            expect_column_values_to_be_between(
                column="price", min_value=1, max_value=1e4, mostly=0.95
            )
        ]
```

Native in-line data expectations & alerts

8. Feature Drift Detection



Realtime distribution for individual features

Coming to OSS Soon!

Thank You!

Nikhil Garg

nikhil@fennel.ai