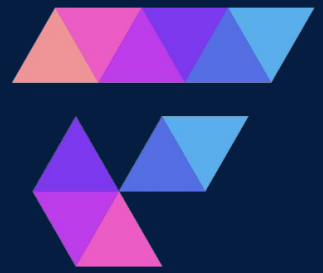


ODSC Europe Feature Engineering With Signal Types

15th June 2023

Feature Engineering with Signal Types

Data is a finite resource, and you want to get the most out of it. So feature engineering becomes vital for machine learning success. Yet we often lose important signals in our data when our data extracts are limited to the same old boring COUNT and SUM aggregations. In this workshop we will apply a new approach to feature engineering ideation, based upon a structure using data semantics and signal types, yet with the freedom to apply domain knowledge.

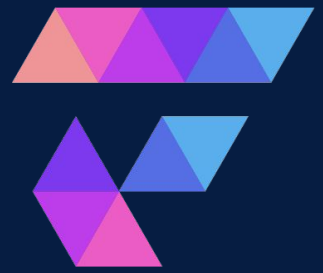
Colin Priest, Chief Evangelist

Before We Commence The Workshop…

- Start Docker
- Open ODSCEuropeWorkshopFeatureEngineeringWithSignalTypes.ipynb
- Run the first Notebook cell

Load the featurebyte library and connect to the local instance of featurebyte

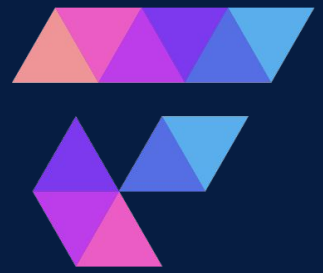
```
# library imports
import pandas as pd
import numpy as np

# load the featurebyte SDK
import featurebyte as fb

# start the local server, then wait for it to be healthy before proceeding
fb.playground()

# this script requires version 0.2.2 or higher
print("FeatureByte Version: " + fb.version)
```



Module 1

Module 1: Goals

- Introduction to the case study
- Understand the concepts of table types, entities, and observation sets
- Identify the primary entity of a use case, feature, and feature list
- Define an observation set for materializing features
- Follow a structured process for feature ideation

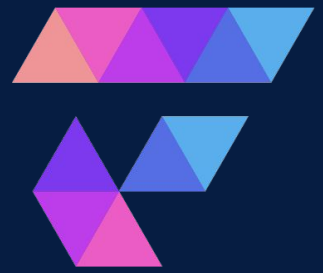


What is FeatureByte?

FeatureByte Open Source SDK

FeatureByte is a free and source available feature platform designed to:

- Create state-of-the-art features, not data pipelines: Create features for Machine Learning with just a few lines of code. Leave the plumbing and pipelining to FeatureByte. We take care of orchestrating the data ops - whether it's time-window aggs or backfilling, so you can deliver more value from data.
- Improve Accuracy through data: Use the intuitive feature declaration framework to transform creative ideas into training data in minutes. Ditch the limitations of ad-hoc pipelines for features with much more scale, complexity and freshness.
- Streamline machine learning data pipelines: Get more value from AI. Faster. Deploy and serve features in minutes, instead of weeks or months. Declare features in Python and automatically generate optimized data pipelines — all using tools you love like Jupyter Notebooks.

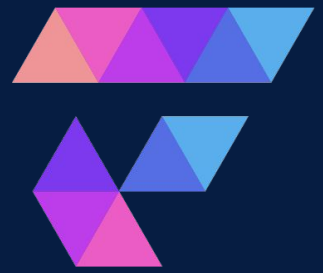


Case Study

Case Study

- A bank provides card cards to its customers
- The bank wants to identify transactions are likely to be fraudulent
- They have historical data about the customers, credit cards, transactions, and fraud

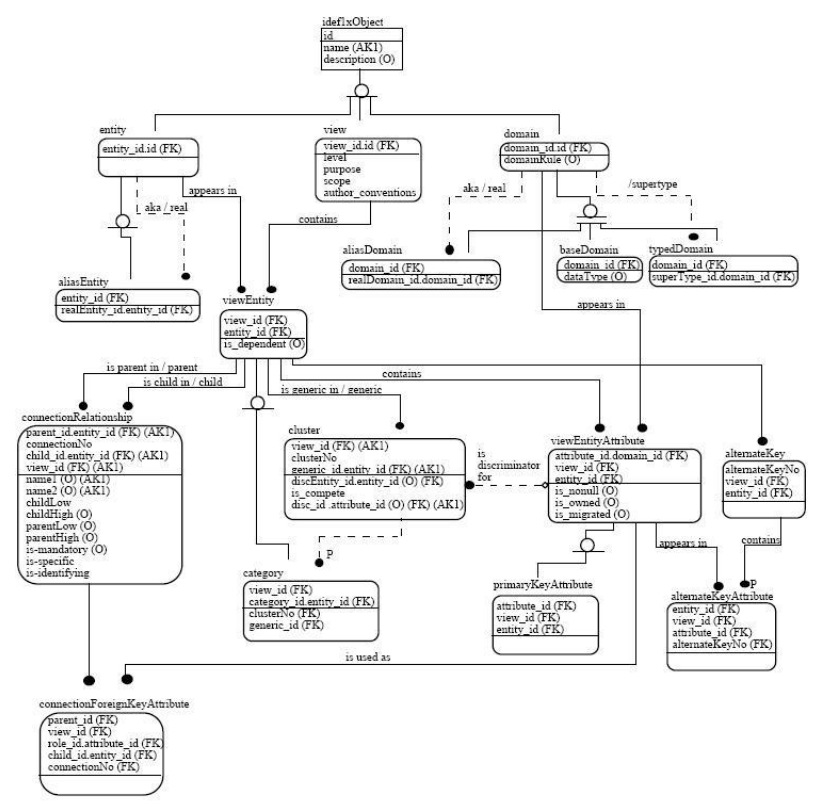
The challenge is to quickly create a diverse yet intuitive range of features that might be helpful in predicting which transactions are fraudulent.



Data Modeling

Data Modeling

- Data semantics should drive feature engineering
- Entities and table relationships determine the feature engineering steps that are possible and/or required e.g whether aggregation is necessary
- Domain knowledge adds value above and beyond data semantics, but its implementation must be coherent with the data model
- This workshop is about feature engineering for signal types. To save time, the data model has already been set up for you



Case Study Data Model

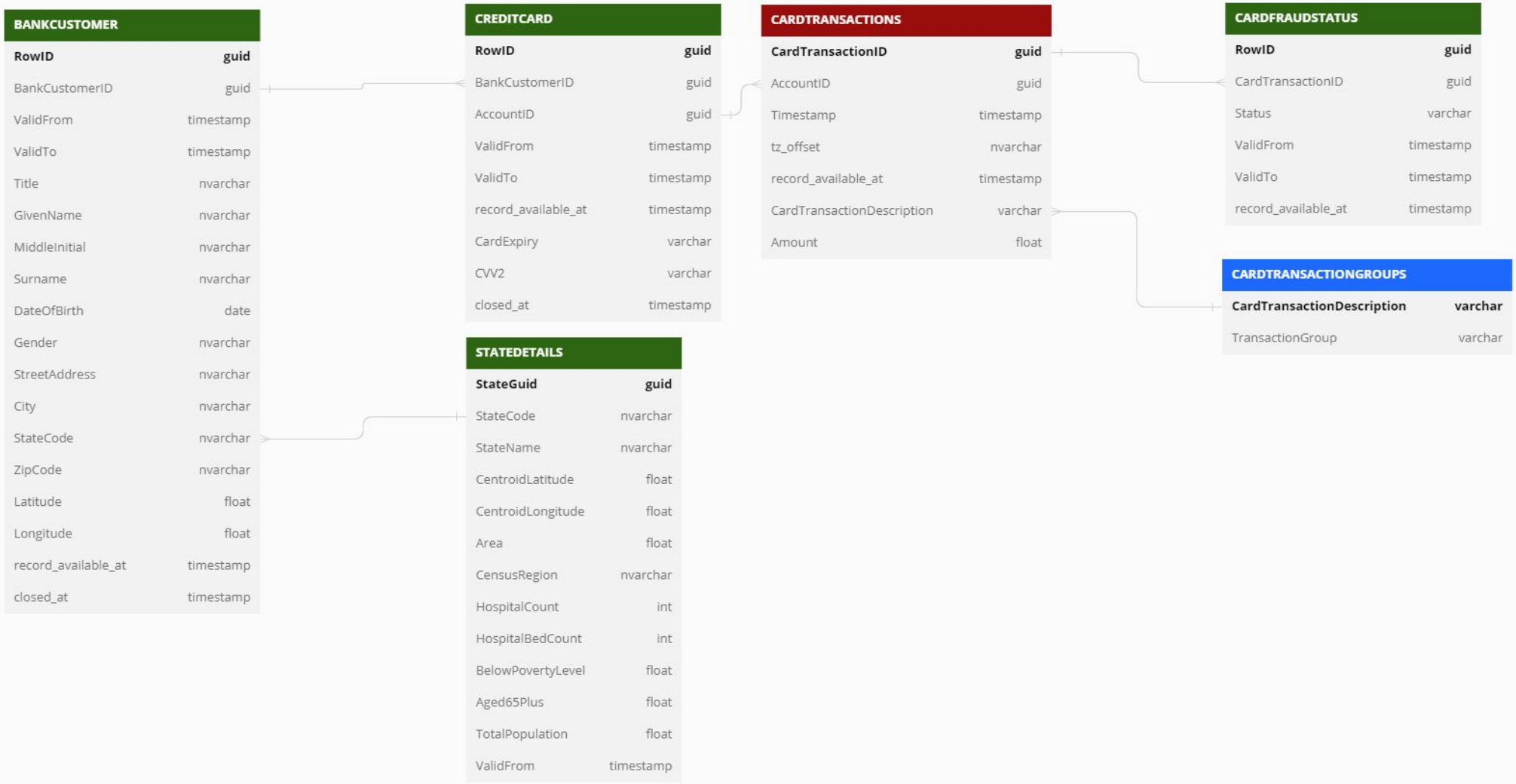
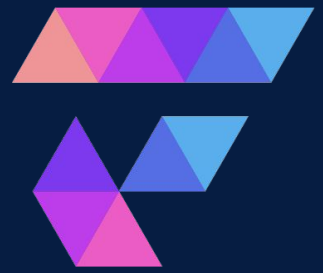


Table Types

- A table’s type will determine the types of feature engineering operations possible on the table's views and enforces guardrails accordingly.
- Currently, FeatureByte recognizes four table types:
  - **Event Table:** a table where each row indicates a unique business event occurring at a particular time.
  - **Item Table:** a table containing detailed information about a specific business event.
  - **Slowly Changing Dimension Table** (SCD): a table containing data that changes slowly and unpredictably over time.
  - **Dimension Table:** a table containing static descriptive data.



Entities

Entities

Use a Variety of Entities

- An entity is a real-world object or concept that is represented by fields in the source tables.
- All features must relate to an entity (or entities) as their primary unit of analysis.
- Table joins add valuable signals.

Parent Entities

- A group that an entity belongs to e.g. gender, or country.
- Can be joined to the unit of analysis as a robust signal source.

Child Entities

- More granular than your unit of analysis.
- Needs to be aggregated.

Entities

List the Entities and Relationships

```
catalog.list_entities()
```

✓ 0.1s

	id	name	serving_names	created_at
0	6477007942f4a7a1236f906b	transaction_group	[TRANSACTIONGROUP]	2023-05-31 08:08:25.698
1	6477007942f4a7a1236f906a	gender	[GENDER]	2023-05-31 08:08:25.557
2	6477007942f4a7a1236f9069	card_transaction_description	[CARDTRANSACTIONDESCRIPTION]	2023-05-31 08:08:25.423
3	6477007942f4a7a1236f9068	card_transaction	[CARDTRANSACTIONID]	2023-05-31 08:08:25.288
4	6477007942f4a7a1236f9067	credit_card	[ACCOUNTID]	2023-05-31 08:08:25.156
5	6477007842f4a7a1236f9066	USA_state	[STATECODE]	2023-05-31 08:08:24.979
6	6477007842f4a7a1236f9065	bank_customer	[BANKCUSTOMERID]	2023-05-31 08:08:24.806

```
catalog.list_relationships()
```

✓ 0.3s

	id	relationship_type	entity	related_entity	relation_table	relation_table_type	enabled	created_at	updated_at
0	6477007b7043bb8a4571ce93	child_parent	card_transaction_description	transaction_group	CARDTRANSACTIONGROUPS	dimension_table	True	2023-05-31 08:08:27.834	None
1	6477007b7043bb8a4571ce89	child_parent	card_transaction	card_transaction_description	CARDTRANSACTIONS	event_table	True	2023-05-31 08:08:27.265	None
2	6477007b7043bb8a4571ce84	child_parent	card_transaction	credit_card	CARDTRANSACTIONS	event_table	True	2023-05-31 08:08:27.128	None
3	6477007a7043bb8a4571ce7d	child_parent	credit_card	bank_customer	CREDITCARD	scd_table	True	2023-05-31 08:08:26.862	None
4	6477007a7043bb8a4571ce74	child_parent	bank_customer	gender	BANKCUSTOMER	scd_table	True	2023-05-31 08:08:26.135	None
5	647700797043bb8a4571ce6f	child_parent	bank_customer	USA_state	BANKCUSTOMER	scd_table	True	2023-05-31 08:08:25.998	None

Primary Entities

- The primary entity of a feature defines the level of analysis for that feature.
- The primary entity of a feature list determines the entities that can be used to serve the feature list, which typically corresponds to the primary entity of the Use Case that the feature list was created for.

List the Entities and Relationships

```
catalog.list_entities()
```

✓ 0.1s

	id	name	serving_names	created_at
0	6477007942f4a7a1236f906b	transaction_group	[TRANSACTIONGROUP]	2023-05-31 08:08:25.698
1	6477007942f4a7a1236f906a	gender	[GENDER]	2023-05-31 08:08:25.557
2	6477007942f4a7a1236f9069	card_transaction_description	[CARDTRANSACTIONDESCRIPTION]	2023-05-31 08:08:25.423
3	6477007942f4a7a1236f9068	card_transaction	[CARDTRANSACTIONID]	2023-05-31 08:08:25.288
4	6477007942f4a7a1236f9067	credit_card	[ACCOUNTID]	2023-05-31 08:08:25.156
5	6477007842f4a7a1236f9066	USA_state	[STATECODE]	2023-05-31 08:08:24.979
6	6477007842f4a7a1236f9065	bank_customer	[BANKCUSTOMERID]	2023-05-31 08:08:24.806

Entities and Feature Ideation

- Features may be on parent entities, not just the primary entity
- Fraudsters may target customers by both their attributes and behaviors
- Fraudsters may target geographies
- Fraudsters may have recognizable behaviours

List the Entities and Relationships

```
catalog.list_entities()
```

✓ 0.1s

	id	name	serving_names	created_at
0	6477007942f4a7a1236f906b	transaction_group	[TRANSACTIONGROUP]	2023-05-31 08:08:25.698
1	6477007942f4a7a1236f906a	gender	[GENDER]	2023-05-31 08:08:25.557
2	6477007942f4a7a1236f9069	card_transaction_description	[CARDTRANSACTIONDESCRIPTION]	2023-05-31 08:08:25.423
3	6477007942f4a7a1236f9068	card_transaction	[CARDTRANSACTIONID]	2023-05-31 08:08:25.288
4	6477007942f4a7a1236f9067	credit_card	[ACCOUNTID]	2023-05-31 08:08:25.156
5	6477007842f4a7a1236f9066	USA_state	[STATECODE]	2023-05-31 08:08:24.979
6	6477007842f4a7a1236f9065	bank_customer	[BANKCUSTOMERID]	2023-05-31 08:08:24.806

Observation Sets

Observation Sets

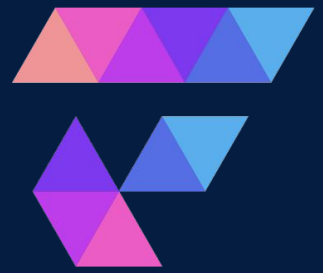
- FeatureByte minimizes data movement
- So it uses feature declarations without materialization of values
- However, materialization is required to
  - sensibility check new feature declarations
  - create AI-ready training data
  - serve AI-ready inference data
- Materialization requires a list of entity keys and points in time, supplied in an observation set
- This workshop uses short observation sets to view sample values of newly declared features

Code: Observation Sets

observation set data are sourced from views

```
sample_transactions = card_transactions_view[card_transactions_view.Timestamp.dt.year == 2022].create_observation_table(
    name="10 Random Credit Card Transaction IDs during 2022",
    sample_rows=10,
    columns=["Timestamp", "CardTransactionID"],
    columns_rename_mapping={
        "Timestamp": "POINT_IN_TIME",
        "CardTransactionID": "CARDTRANSACTIONID",
    }
).to_pandas()

# sort the observations by timestamp
sample_transactions = sample_transactions.sort_values(by="POINT_IN_TIME")
```

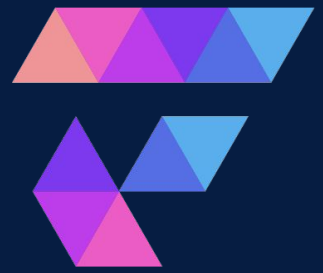
an observation set contains pairs of entity keys and points in time



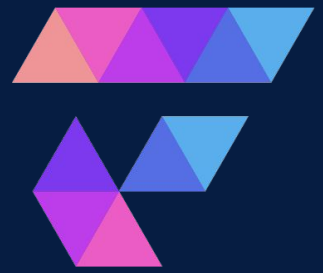
Feature Ideation

Ideation Process

- Signal Type
  - will be covered in module 2
- Entities:
  - which entities are relevant to your use case?
- Attributes:
  - what attributes or events of that entity are likely to have a useful signal
- Ideas:
  - hypothesis about the real world
  - metric



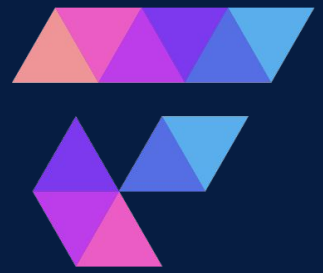
Q&A



Module 2

Module 2: Goals

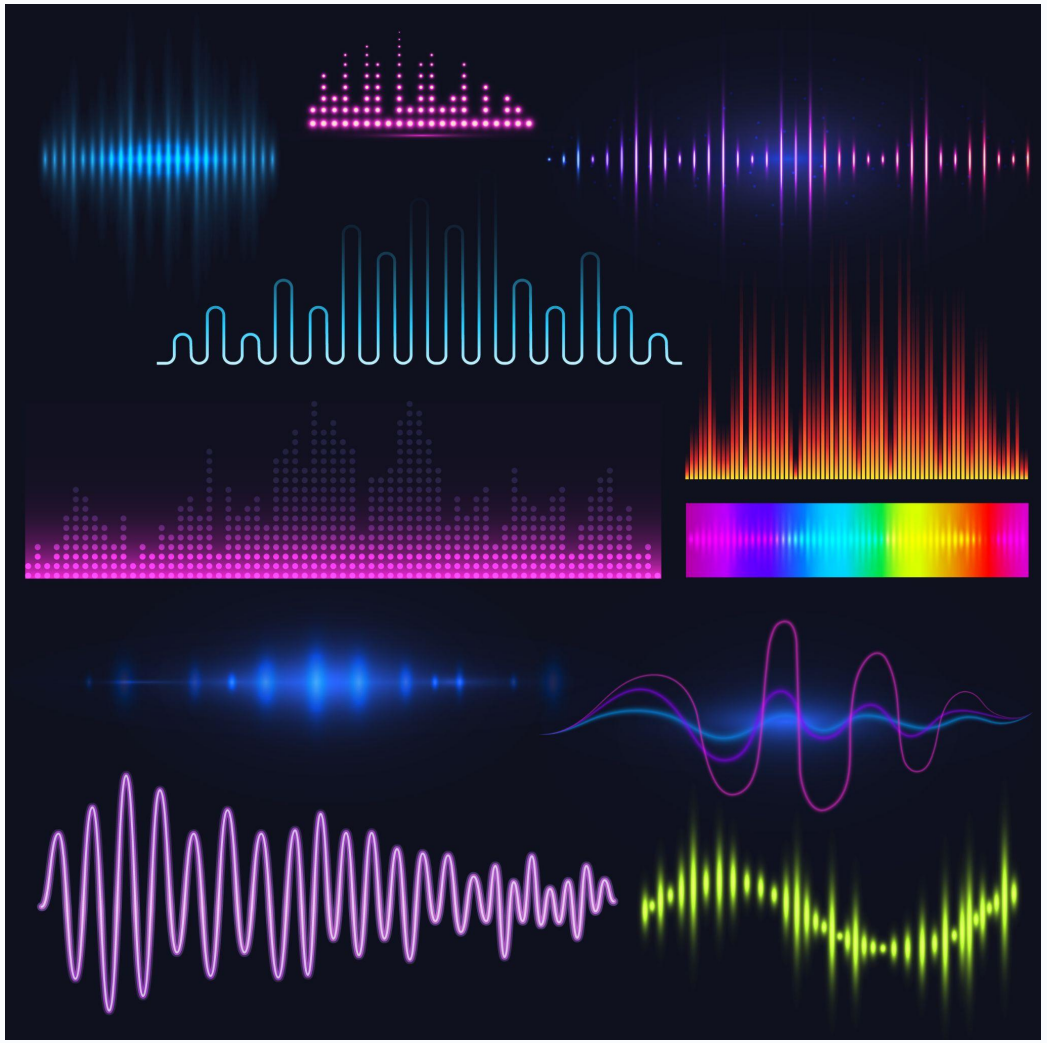
- List 13 different signal types
- Follow a feature ideation process for each signal type
- Declare features using FeatureByte
- Materialize feature values



Signal Types

Signal Types

- A key purpose of feature engineering is to extract signals from data
- The concept of signals is not new. RFM analysis has been used in marketing since 1995
- But there are many more signal types beyond RFM. In this workshop we will introduce 10 additional signal types.
- When designing signals, it is helpful to make them intuitive and aim for statistical independence



Recency Signals

Attributes of the latest event to occur.

Metrics

- Time since last event.
- Label of last event.
- Magnitude of last event.

Examples

- How much did a customer spend on their last purchase?
- How long since the last equipment failure?
- Was a patient's last visit for an emergency?

Ideation: Recency Signals and Fraud

- Entities:
  - customer
  - transactions
- Attributes:
  - transaction description
  - elapsed time
- Ideas:
  - fraudsters may return after recently targeting the customer - time since customer's last fraud
  - latest fraud trends - description of last fraudulent across all customers

Coding: Recency and Fraud

convert slow changing dimension data to events whenever the fraud status changes

filter to keep only the original fraud report

Create events from an SCD

```
# create a change view on the fraud status, that captures fraud status changes
card_fraud_status_change_view = card_fraud_status_table.get_change_view(
    track_changes_column="Status",
)

# filter for only confirmed fraud
card_fraud_status_change_view_confirmed = card_fraud_status_change_view[card_fraud_status_change_view.new_Status == "Reported"].copy()

# join the transactions view to get the transaction details
card_fraud_status_change_view_confirmed = card_fraud_status_change_view_confirmed.join(
    card_transactions_view[["CardTransactionID", "Timestamp", "tz_offset", "AccountID", "BankCustomerID", "CardTransactionDescription", "Amount"]],
)

card_fraud_status_change_view_confirmed.sample()
```

join the transaction details

Coding: Recency Signals and Fraud

group by customer

use LATEST aggregation over a time window

How much time has elapsed since the customer's last fraudulent transaction report?

```
# get the latest date for each customer
customer_latest_fraud = card_fraud_status_change_view_confirmed.groupby(
    "BankCustomerID"
).aggregate_over(
    "Timestamp",
    method=fb.AggFunc.LATEST,
    feature_names=["Customer Latest Fraud Date"],
    windows=['365d']
)

# get the time since that date
days_since_customer_latest_fraud = (RequestColumn.point_in_time() - customer_latest_fraud["Customer Latest Fraud Date"]).dt.day
days_since_customer_latest_fraud.name = "Days Since Customer Latest Fraud"
customer_latest_fraud["Days Since Customer Latest Fraud"] = days_since_customer_latest_fraud
```

compare to prediction point in time

Coding: Recency Signals and Fraud

across all entities

event data from a filtered view of fraud reports

What was the most recent fraudulent transaction description across all customers?

```
latest_fraudulent_transaction_description = card_fraud_status_change_view_confirmed.groupby(
    []
).aggregate_over(
    "CardTransactionDescription",
    method=fb.AggFunc.LATEST,
    feature_names=["Latest Fraud Description"],
    windows=['365d']
)
```

time window - return None if no matches within the window

Cross-Aggregation Signals

Grouping the data by a specific entity and aggregating a column's values across categories of a categorical column

Metrics

- Counts of item types
- Sum of values of item types
- Max values of item types
- Latest attribute of item types

Examples

- What are the counts of each item type in a shopping basket?
- What is the total value of item type for each customer’s shipping orders over 30 days?
- What is the maximum weight of each type of item in a shipping order?

Ideation: Cross-Aggregation and Fraud

- Entities:
  - customer
  - transactions
- Attributes:
  - transaction description
- Ideas:
  - customers' spending patterns could make them targets - total spend by description
  - typical descriptions of fraudulent transactions - count by description

Coding: Cross-Aggregation and Fraud

cross-categorized by transaction description

use SUM aggregation over a time window

```
# create a cross aggregation of the transaction amount by customer and description over the past 28 days
customer_transaction_descriptions = card_transactions_view.groupby(
    "BankCustomerID",
    category="CardTransactionDescription"
).aggregate_over(
    "Amount",
    method=fb.AggFunc.SUM,
    feature_names=["Total Amount by Customer and Description"],
    windows=['28d']
)["Total Amount by Customer and Description"]

# create a feature that is the highest category
customer_highest_spend_category = customer_transaction_descriptions.cd.most_frequent()
customer_highest_spend_category.name = "Customer Highest Spend Category"
```

the label associated with the highest spend

Coding: Cross-Aggregation and Fraud

across all fraudulent transactions

use COUNT aggregation over a time window

What was the most common transaction description for fraudulent transactions over the past 60 days?

```
# create a cross aggregation of the transaction amount by description for fraudulent transactions over the past 60 days
fraudulent_transaction_description = card_fraud_status_change_view_confirmed.groupby(
    [],
    category="CardTransactionDescription"
).aggregate_over(
    None,
    method=fb.AggFunc.COUNT,
    feature_names=["Fraud Total Amount by Description"],
    windows=['60d']
)["Fraud Total Amount by Description"]

# create a feature that is the highest category
fraudulent_most_frequent_category = fraudulent_transaction_description.cd.most_frequent()
fraudulent_most_frequent_category.fillna("-")
fraudulent_most_frequent_category.name = "Fraudulent Most Frequent Category"
```

the most frequent description for fraudulent transactions

Similarity Signals

Compare an individual entity’s attributes or past activity to a group

Metrics

- Ratio of one entity's numeric value to the average or maximum of values over a group of entities.
- Cosine similarity of two entities’ cross-aggregations

Examples

- How similar is one customer's purchases to customers of the same age and gender?
- What is the ratio of a network antenna's maximum temperature to that of all other antennas that are the same model, or located in the same geography?

Ideation: Similarity and Fraud

- Entities:
  - customer
  - transactions
  - state
- Attributes:
  - transaction description
  - transaction amounts
- Ideas:
  - similarity of amount or description to recent fraudulent transactions
  - fraudsters may target a geography - similarity of customer to their geographic neighbours

Coding: Similarity and Fraud

set the frequency to zero if no match

Is the transaction description a match to recent fraudulent transactions from all customers?

```
# get a count of transaction descriptions from recent fraudulent transactions
recent_fraudulent_transaction_description_counts = card_fraud_status_change_view_confirmed.groupby(
    [],
    category="CardTransactionDescription"
).aggregate_over(
    None,
    method=fb.AggFunc.COUNT,
    feature_names=["Recent Fraudulent Transaction Descriptions 14d"],
    windows=['14d']
)["Recent Fraudulent Transaction Descriptions 14d"]

# create a featur that is the transaction description
transaction_description = card_transactions_view.CardTransactionDescription.as_feature("Transaction Description")

# create a feature that flags whether the single transaction has a description that matches any recent fraudulent transaction descriptions
matches_recent_fraudulent_transaction_description = recent_fraudulent_transaction_description_counts.cd.get_relative_frequency(transaction_description)
matches_recent_fraudulent_transaction_description.fillna(0)
matches_recent_fraudulent_transaction_description.name = "Matches Recent Fraudulent Transaction Description"
```

How often does a fraudulent transaction description match this transaction?

Coding: Similarity and Fraud

Is the transaction amount similar to recent fraudulant transactions?

```
# create a feature that is the Z-score of the transaction amount versus recent fraudulent transactions

x = card_transactions_view.Amount.as_feature("Transaction Amount")

mu = card_fraud_status_change_view_confirmed.groupby(
    []
).aggregate_over(
    "Amount",
    method=fb.AggFunc.AVG,
    feature_names=["Mean Fraudulent Transaction Amount 28d"],
    windows=['28d']
)["Mean Fraudulent Transaction Amount 28d"]

sigma = card_fraud_status_change_view_confirmed.groupby(
    []
).aggregate_over(
    "Amount",
    method=fb.AggFunc.STD,
    feature_names=["Std Fraudulent Transaction Amount 28d"],
    windows=['28d']
)["Std Fraudulent Transaction Amount 28d"]

z_score_vs_fraud_28d = (x - mu) / sigma
z_score_vs_fraud_28d.fillna(0)
z_score_vs_fraud_28d.name = "Z-Score of Transaction Amount vs Recent Fraudulent Transactions 28d"

# display sample values
z_score_vs_fraud_28d.preview(
    sample_transactions_with_fraud
)
```

columns can be converted to features

Z-score of the transaction amount versus fraudulent transactions

Coding: Similarity and Fraud

different entities, same aggregation

Create a feature that compares a customer's recent transaction descriptions to those of customers who live in the same geographic region

```
# get the customer's transaction descriptions for the past 28 days
customer_transaction_descriptions_28d = card_transactions_view.groupby(
    "BankCustomerID",
    category="CardTransactionDescription"
).aggregate_over(
    None,
    method=fb.AggFunc.COUNT,
    feature_names=["Customer Transaction Descriptions 28d"],
    windows=['28d']
)["Customer Transaction Descriptions 28d"]

# get the state total's transaction descriptions for the past 28 days
state_transaction_descriptions_28d = card_transactions_view.groupby(
    "StateCode",
    category="CardTransactionDescription"
).aggregate_over(
    None,
    method=fb.AggFunc.COUNT,
    feature_names=["State Transaction Descriptions 28d"],
    windows=['28d']
)["State Transaction Descriptions 28d"]

# get the cosine similarity between the customer's transaction descriptions and the state's transaction descriptions
customer_state_similarity_28d = customer_transaction_descriptions_28d.cd.cosine_similarity(state_transaction_descriptions_28d)
customer_state_similarity_28d.fillna(0)
customer_state_similarity_28d.name = "Customer State Similarity 28d"
```

cosine similarity of cross-aggregations of events

Stability Signals

Compare recent data values and events to those of earlier periods

Metrics

- Ratio of latest numeric value to the average or max of values over a historical time window.
- Cosine similarity of two cross-aggregations from different time periods

Examples

- Are the contents of the latest shopping basket similar to past purchases?
- Is the count or sum of numeric event data over the most recent period similar to the past?
- Is a recent data value anomalous?

Ideation: Stability and Fraud

- Entities:
  - customer
  - transactions
- Attributes:
  - transaction description
  - transaction amounts
- Ideas:
  - fraudulent transactions may look anomalous - similarity of amount to customer's non-fraudulent transactions
  - customer may have changed behavior, triggering fraud - similarity of recent transactions to earlier transactions

Coding: Stability and Fraud

Is the transaction amount similar to the customer's past transactions?

```
x = card_transactions_view.Amount.as_feature("Transaction Amount")

mu = card_transactions_view.groupby(
    "BankCustomerID"
).aggregate_over(
    "Amount",
    method=fb.AggFunc.AVG,
    feature_names=["customer mean transaction amount 365d"],
    windows=['365d']
)["customer mean transaction amount 365d"]

sigma = card_transactions_view.groupby(
    "BankCustomerID"
).aggregate_over(
    "Amount",
    method=fb.AggFunc.STD,
    feature_names=["customer stdev transaction amount 365d"],
    windows=['365d']
)["customer stdev transaction amount 365d"]

transaction_z_score = (x - mu) / sigma
transaction_z_score.name = "Transaction Z Score 365d"
```

Z score of transaction versus history

Coding: Stability and Fraud

same entities, different windows

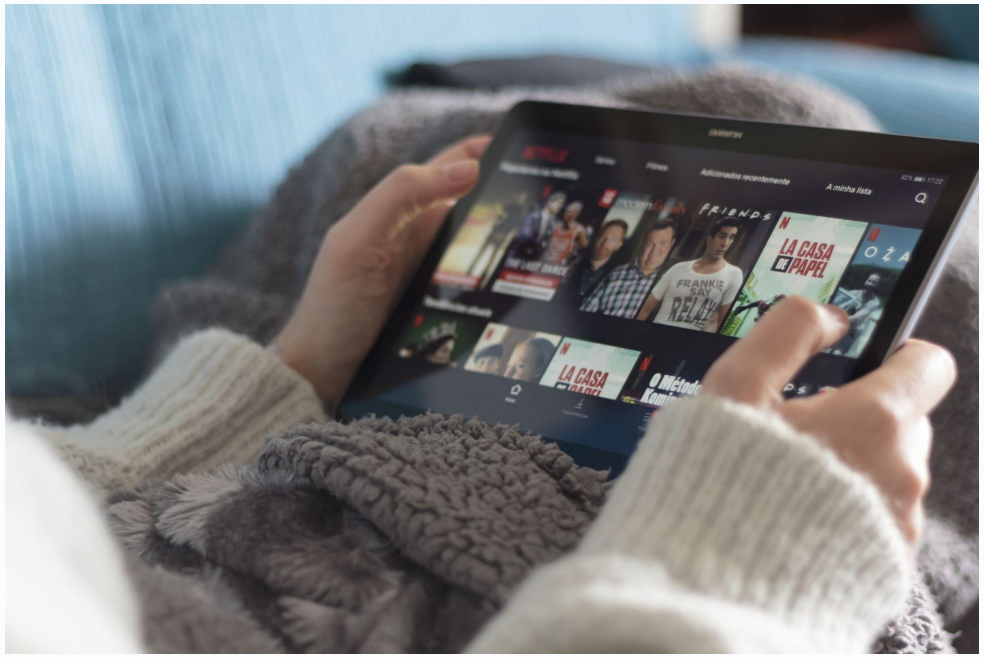
Are the customer's recent transaction descriptions similar to their earlier transactions?

```
# get transactions from the past week and the past year
customer_transaction_descriptions_1w1yr = card_transactions_view.groupby(
    "BankCustomerID",
    category="CardTransactionDescription"
).aggregate_over(
    None,
    method=fb.AggFunc.COUNT,
    feature_names=["customer transaction desc 1w", "customer transaction desc 1yr"],
    windows=['7d','365d']
)

# create a feature from the cosine similarity between the two windows
transaction_count_cosine_similarity = customer_transaction_descriptions_1w1yr["customer transaction desc 1w"].cd.cosine_similarity(
    customer_transaction_descriptions_1w1yr["customer transaction desc 1yr"]
)
transaction_count_cosine_similarity.name = "Customer Transaction Desc Stability 1w1yr"
```

cosine similarity of cross-aggregations of events

Clumpiness Signals

The occurrence of long gaps in time between bursts of multiple events



Metrics

- Entropy of inter-event times
- Coefficient of variation of inter-event times

Examples

- Which customers are binge-watching TV shows?
- Do equipment failures occur in cascades?
- After a long break without purchasing, does a customer return to a store a few times over several days to purchase items?

Ideation: Clumpiness and Fraud

- Entities:
  - customer
  - transactions
- Attributes:
  - inter-event times
- Ideas:
  - fraudsters may target predictable behaviors - clumpiness of IETs
  - fraudulent transactions may suddenly occur in a burst of time - clumpiness of IETs

Coding: Clumpiness and Fraud

event table - filtered for purchases only

Calculate the Inter-Event-Times

```
# get the inter-event-times for the customer's transactions
view = purchases_view.copy()
ts_col = view[view.timestamp_column]
view["InterEventTime"] = (ts_col - ts_col.lag("BankCustomerID")).dt.day
view.preview()
```

lagged event times, grouped by customer

Coding: Clumpiness and Fraud

Does the customer exhibit binge shopping behaviours or erratic timing of purchases?

```
# calculate the entropy and coefficient of variation of the inter-event-times
windows = ["28d"]

# we can easily create inter-event times
view = card_transactions_view.copy()
ts_col = view[view.timestamp_column]
view["IET"] = (ts_col - ts_col.lag("BankCustomerID")).dt.day

# from the inter-event times we can transform and aggregate to get IET clumpiness and diversity metrics
a = view["IET"]
a[a < 0.001] = 0.001
view["a * log(a)"] = a * a.log()

clumpiness_features = fb.FeatureGroup([])
for window in windows:
    b = view.groupby("BankCustomerID").aggregate_over(
        "IET",
        method=fb.AggFunc.SUM,
        windows=[window],
        feature_names=[f"sum(a){window}"],
    )[f"sum(a){window}"]
    # handle zeroes
    b = b.astype(float)
    b[b < 1e-8] = 1e-8

    mu = view.groupby("BankCustomerID").aggregate_over(
        "IET",
        method=fb.AggFunc.AVG,
        windows=[window],
        feature_names=[f"sum(a){window}"],
    )[f"sum(a){window}"]
    # handle zeroes
    mu = mu.astype(float)
    mu[mu < 1e-8] = 1e-8

    # clumpiness - entropy
    entropy = view.groupby("BankCustomerID").aggregate_over(
        "a * log(a)",
        method=fb.AggFunc.SUM,
        windows=[window],
        feature_names=[f"sum(a * log(a)){window}"],
        )[f"sum(a * log(a)){window}"] * -1 / b + mu.log()
    entropy.fillna(0)
    clumpiness_features[f"Card Transactions Timing Clumpiness Entropy (zero corrected)_({window})"] = entropy

    # regularity - coefficient of variation
    cv = view.groupby("BankCustomerID").aggregate_over(
        "IET",
        method=fb.AggFunc.STD,
        windows=[window],
        feature_names=[f"std(a){window}"],
    )[f"std(a){window}"] / mu
    cv.fillna(0)
    clumpiness_features[f"Card Transactions Timing Coefficient of Variation (zero corrected)_({window})"] = cv
```

✓ 37.2s

entropy

coefficient of variation

Diversity Signals

How variable are the data values?

Metrics

- Coefficient of variation of amounts
- Entropy of labels

Examples

- How variable are the monthly invoice amounts over the past 6 months?
- How stable is a patient's blood pressure?
- Does a customer always purchase similar products?

Ideation: Diversity and Fraud

- Entities:
  - customer
  - transactions
- Attributes:
  - transaction descriptions
  - transaction amounts
- Ideas:
  - fraudsters may target predictable behaviors - diversity of transactions
  - customers with diverse spending habits are more likely to have their details stolen - diversity of transactions

Coding: Diversity and Fraud

How variable are a customer's transaction amounts?

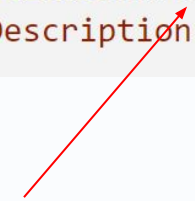
```
# mean transaction amount by customer over 28 days
mean_transaction_amount_28d = card_transactions_view.groupby(
    "BankCustomerID"
).aggregate_over(
    "Amount",
    method=fb.AggFunc.AVG,
    feature_names=["customer mean transaction amount 28d"],
    windows=['28d']
)["customer mean transaction amount 28d"]

# standard deviatio of transaction amount by customer over 28 days
stdev_transaction_amount_28d = card_transactions_view.groupby(
    "BankCustomerID"
).aggregate_over(
    "Amount",
    method=fb.AggFunc.STD,
    feature_names=["customer stdev transaction amount 28d"],
    windows=['28d']
)["customer stdev transaction amount 28d"]

# create a feature from the ratio of the standard deviation to the mean
transaction_amount_diversity_28d = stdev_transaction_amount_28d / mean_transaction_amount_28d
transaction_amount_diversity_28d.name = "Customer Transaction Amount Diversity 28d"
```

coefficient of variation is a measure of numeric diversity

Coding: Diversity and Fraud

How varied are a customers transaction descriptions?

```
# create a cross aggregation feature for transaction descriptions by customer id
customer_transaction_descriptions = card_transactions_view.groupby(
    "BankCustomerID",
    category="CardTransactionDescription"
).aggregate_over(
    None,
    method=fb.AggFunc.COUNT,
    feature_names=["customer transaction desc 28d"],
    windows=['28d']
)["customer transaction desc 28d"]

# create a feature that is the entropy of the transaction descriptions
transaction_description_diversity_28d = customer_transaction_descriptions.cd.entropy()
transaction_description_diversity_28d.name = "Customer Transaction Description Diversity 28d"
```



entropy is a measure of categorical diversity

Change Signals

The occurrence or magnitude of changes to slowly changing attributes

Metrics

- Did an attribute change?
- How many times did an attribute change?
- By how much did it change?

Examples

- Did a customer change address over the past year? How far?
- Has a password been reset in the past 48 hours?
- Was network equipment replaced with a different make or model?

Ideation: Diversity and Fraud

- Entities:
  - customer
  - credit card
- Attributes:
  - count of changes
- Ideas:
  - fraudsters may target customers who change address
  - fraudsters may target or intercept the delivery of new credit cards

Coding: Change and Fraud

Has the customer changed address over the past year?

```
# get a change view from the bank customer table
bank_customer_change_view = bank_customer_table.get_change_view(
    track_changes_column="StreetAddress",
)

# filter out the first address entered for each customer
bank_customer_change_view = bank_customer_change_view[bank_customer_change_view.past_StreetAddress.notnull()]

bank_customer_change_view.preview()
```

create events whenever an address changes

```
# count the number of times the customer has changed their address
customer_address_changes = bank_customer_change_view.groupby(
    "BankCustomerID"
).aggregate_over(
    None,
    method=fb.AggFunc.COUNT,
    feature_names=["customer address changes 365d"],
    windows=['365d']
)["customer address changes 365d"]
```

use None when counting events

Coding: Change and Fraud

create events whenever a card expiry month changes

Has the customer been issued a new credit card in the past 3 months?

```
# get a change view from the credit card table
credit_card_change_view = credit_card_table.get_change_view(
    track_changes_column="CardExpiry",
)

# join the credit card view to get the customer ID
credit_card_change_view = credit_card_change_view.join(
    credit_card_view[["AccountID", "BankCustomerID"]]
)

# count the number of times a new card has been issued over the past 90 days
card_issued_90d = credit_card_change_view.groupby(
    "BankCustomerID"
).aggregate_over(
    None,
    method=fb.AggFunc.COUNT,
    feature_names=["card issued 90d"],
    windows=['90d']
)["card issued 90d"]
```

Timing Signals

The regularity or seasonality of events

Metrics

- Entropy of day of the week of events.
- Cross-aggregation of events by hour or month

Examples

- Does a customer always shop on the same days of the week?
- Do equipment failures tend to occur at different times of the day or times of year?

Ideation: Timing and Fraud

- Entities:
  - customer
  - transaction
- Attributes:
  - hour of day, day of week
- Ideas:
  - fraudsters may operate at unusual hours
  - fraudsters may target customers with regular behaviors throughout the day or week

Coding: Timing and Fraud

create a new column in a view, then create a feature from it

What hour of the day did the transaction occur?

```
purchases_view["hour"] = purchases_view.Timestamp.dt.hour
transaction_hour = purchases_view.hour.as_feature("transaction hour")
```

How frequently does the customer shop this hour of the day?

```
# do a cross aggregation of hour of the day of transactions over 90 days grouped by customer
customer_transaction_hour = purchases_view.groupby(
    "BankCustomerID",
    category="hour"
).aggregate_over(
    None,
    method=fb.AggFunc.COUNT,
    feature_names=["customer transaction hour 90d"],
    windows=['90d']
)["customer transaction hour 90d"]

# what proportion of the transactions were made at this hour
customer_transaction_hour_frequency = customer_transaction_hour.cd.get_relative_frequency(transaction_hour)
customer_transaction_hour_frequency.name = "customer transaction hour frequency 90d"
```

cross-aggregation to see how frequently the customer has transactions at this time of day

Coding: Timing and Fraud

create a new column in a view, then cross-aggregate

Does the customer tend to shop on the same days of each the week?

```
# do a cross aggregation of the day of week of transactions over 90 days grouped by customer
purchases_view["weekday"] = purchases_view.Timestamp.dt.day_of_week
customer_transaction_day_of_week = purchases_view.groupby(
    "BankCustomerID",
    category="weekday"
).aggregate_over(
    None,
    method=fb.AggFunc.COUNT,
    feature_names=["customer transaction day of week 90d"],
    windows=['90d']
)["customer transaction day of week 90d"]

# make a feature that is the entropy of the day of week of transactions
transaction_day_of_week_entropy = customer_transaction_day_of_week.cd.entropy()
transaction_day_of_week_entropy.name = "Customer Transaction Day of Week Entropy 90d"
```

use entropy to measure the diversity of days of the week

Location Signals

The static or dynamic location of an event or entity.

Metrics

- The latitude and longitude of an event.
- Distance between locations, or distance moved within a time period.

Examples

- What is the distance between the shop and the customer address?
- Is the shipping address in a remote location? What is the population density in that state?
- How far has a vehicle moved in the past hour?

Ideation: Location and Fraud

- Entities:
  - customer
  - state
- Attributes:
  - latitude / longitude
  - distance
- Ideas:
  - fraudsters may target customers by location, even within a state

Coding: Location and Fraud

What is the latitude and longitude of the customers' address?

```
customer_latitude = bank_customer_view.Latitude.as_feature("customer latitude")
customer_longitude = bank_customer_view.Longitude.as_feature("customer longitude")
```

customer as the entity

create feature from a column

Coding: Location and Fraud

state as the entity

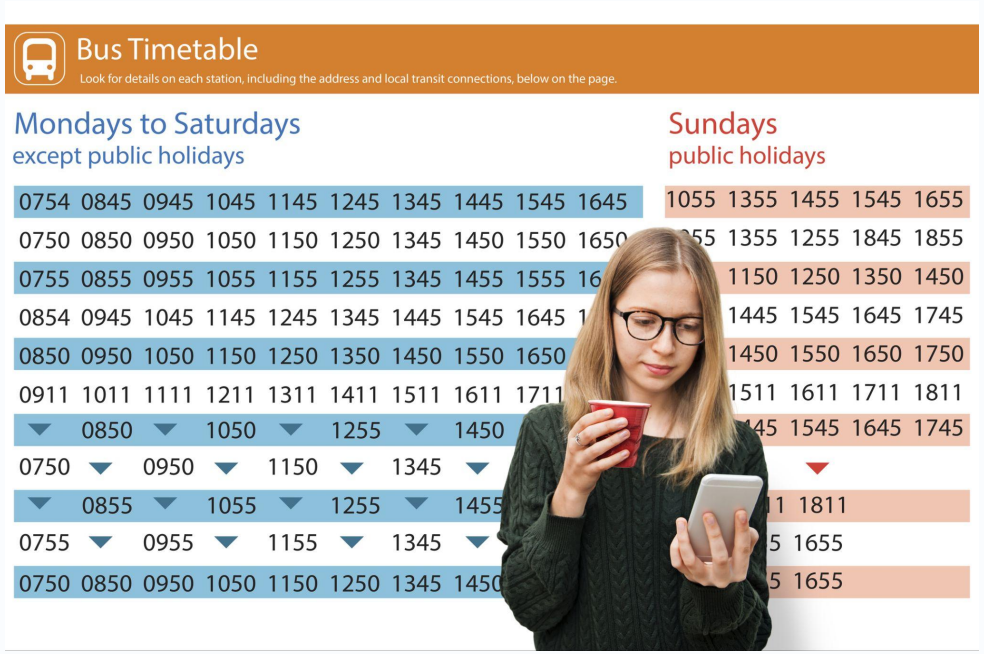
What is the distance between the customer's residence and the state centroid?

```
state_centroid_latitude = state_details_view.CentroidLatitude.as_feature("state centroid latitude")
state_centroid_longitude = state_details_view.CentroidLongitude.as_feature("state centroid longitude")

# create a feature that is the distance between the customer address and the state centroid
customer_state_distance = haversine_distance(
    customer_latitude,
    customer_longitude,
    state_centroid_latitude,
    state_centroid_longitude
)
customer_state_distance.name = "Customer State Distance"
```

no join is required because the new feature is a transformation of four features, with parent/child entity relationships

Frequency Signals

How often do events occur?



Metrics

- Number of events occurring over a time window.

Examples

- How many hospital admissions has the patient had in the past 12 months?
- How many international phone calls has a customer made in the past month?

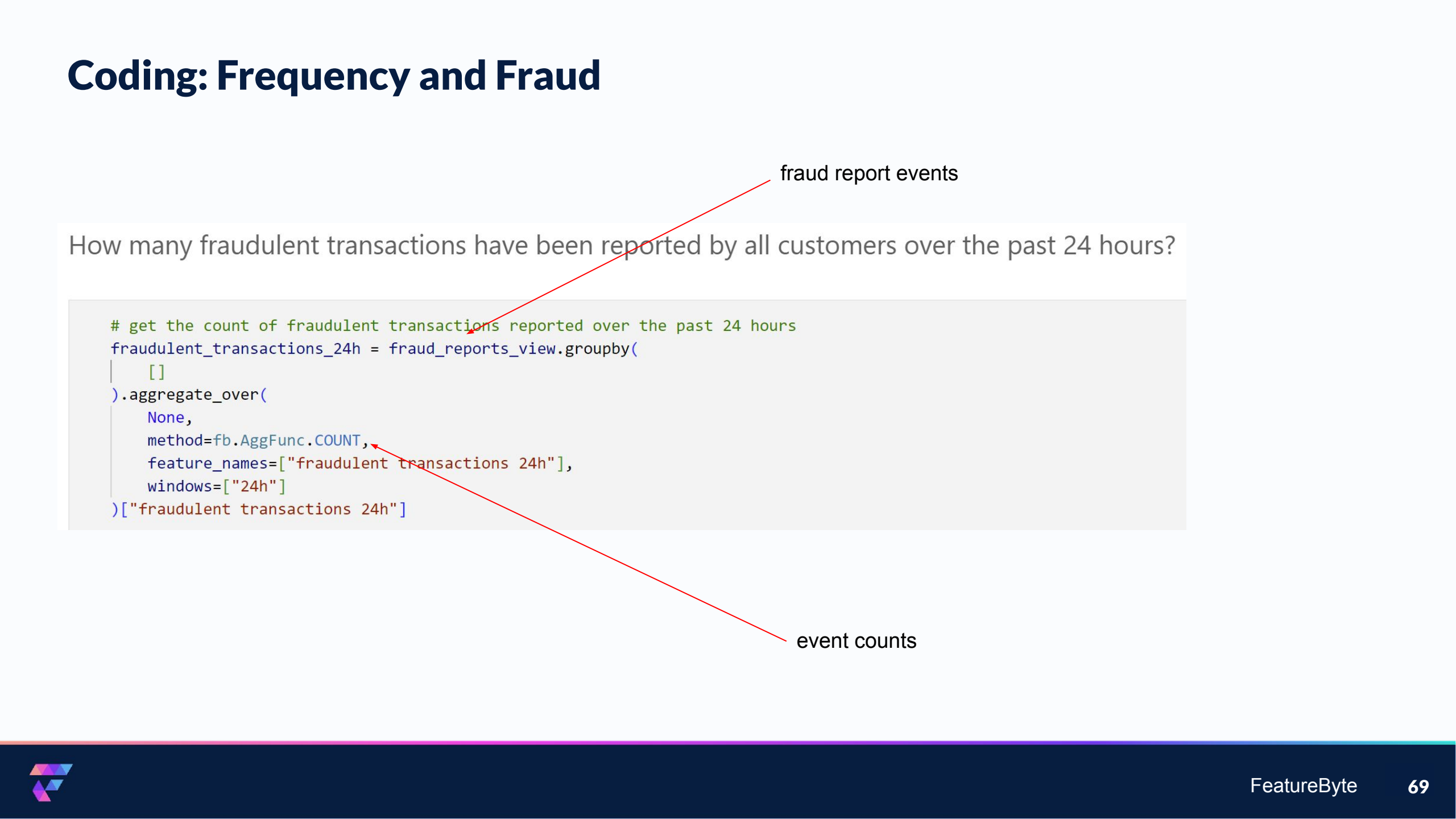
Ideation: Frequency and Fraud

- Entities:
  - customer
  - transaction
- Attributes:
  - event count
- Ideas:
  - currently, maybe there is a heightened level of fraud activity
  - customers with many transactions may be more likely to have their details stolen

Coding: Frequency and Fraud

How many fraudulent transactions have been reported by all customers over the past 24 hours?

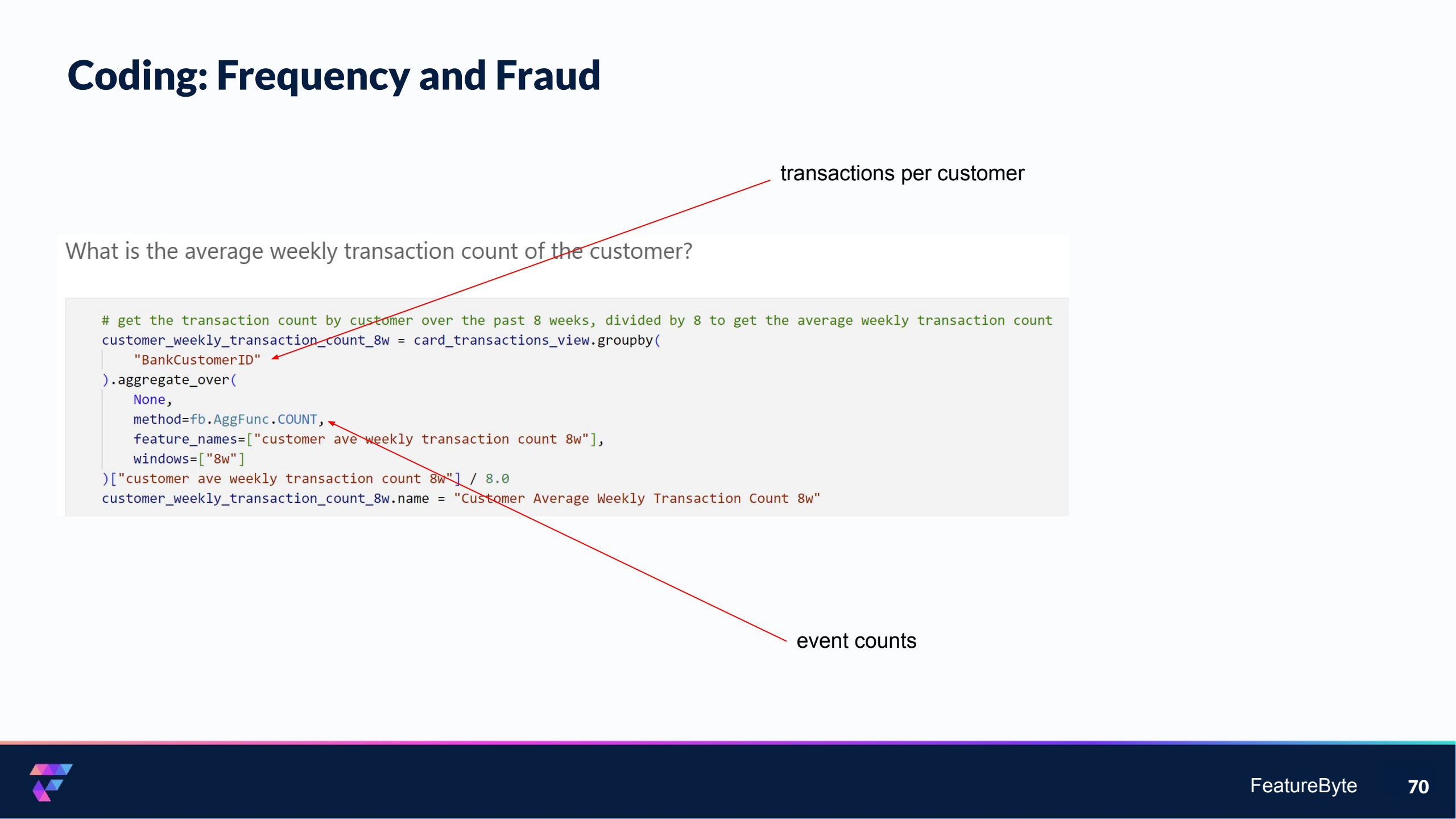
```
# get the count of fraudulent transactions reported over the past 24 hours
fraudulent_transactions_24h = fraud_reports_view.groupby(
    []
).aggregate_over(
    None,
    method=fb.AggFunc.COUNT,
    feature_names=["fraudulent transactions 24h"],
    windows=["24h"]
)["fraudulent transactions 24h"]
```



Coding: Frequency and Fraud

What is the average weekly transaction count of the customer?

```
# get the transaction count by customer over the past 8 weeks, divided by 8 to get the average weekly transaction count
customer_weekly_transaction_count_8w = card_transactions_view.groupby(
    "BankCustomerID"
).aggregate_over(
    None,
    method=fb.AggFunc.COUNT,
    feature_names=["customer ave weekly transaction count 8w"],
    windows=["8w"]
)["customer ave weekly transaction count 8w"] / 8.0
customer_weekly_transaction_count_8w.name = "Customer Average Weekly Transaction Count 8w"
```



Monetary Signals

What are the monetary amounts of events over a time window?

Metrics

- Total cost of events occurring over a time window.
- Average cost of events occurring over a time window

Examples

- How much did a customer spend over the past 12 months?
- What was the average discount applied to a customer's purchases over the past month?

Ideation: Monetary and Fraud

- Entities:
  - customer
  - transaction
- Attributes:
  - transaction amount
- Ideas:
  - fraudsters may target customers by their socio-economic status - can use transaction amounts or interest charges as a proxy
  - on average, fraudulent transactions may be larger or smaller than typical

Coding: Monetary and Fraud

filtered views for specific types of transactions

What is the average weekly purchases expenditure of the customer?

```
# get the sum of transaction amounts over the past 8 weeks, then divide by 8 to get the average weekly transaction total
customer_weekly_purchases_total_8w = purchases_view.groupby(
    "BankCustomerID"
).aggregate_over(
    "Amount",
    method=fb.AggFunc.SUM,
    feature_names=["customer ave weekly purchases total 8w"],
    windows=["8w"]
)["customer ave weekly purchases total 8w"] / 8.0
customer_weekly_purchases_total_8w.name = "Customer Average Weekly Purchases Total 8w"
```

What are the total interest charges to the customer over the past 30 days?

```
# get the sum of interest charges over the past 30 days, grouped by bank customer id
customer_interest_charges_30d = interest_payments_view.groupby(
    "BankCustomerID"
).aggregate_over(
    "Amount",
    method=fb.AggFunc.SUM,
    feature_names=["customer interest charges 30d"],
    windows=["30d"]
)["customer interest charges 30d"]
```

monetary amounts

Coding: Monetary and Fraud

filtered view for a specific type of transactions

What is the mean amount of fraudulent transactions over the past 4 weeks?

```
# get the sum of interest charges over the past 4 weeks, grouped by bank customer id
mean_fraud_transaction_amount_4w = fraud_reports_view.groupby(
    "BankCustomerID"
).aggregate_over(
    "Amount",
    method=fb.AggFunc.AVG,
    feature_names=["mean fraud transaction amount 4w"],
    windows=["30d"]
)["mean fraud transaction amount 4w"]
```

monetary amounts

Lookup Signals

A lookup feature is an entity's point-in-time attribute value that rarely or never changes

Metrics

- Scalar value e.g. label

Examples

- Customer date of birth
- Product category

Ideation: Lookup and Fraud

- Entities:
  - customer
  - transaction
- Attributes:
  - age
  - transaction group label
- Ideas:
  - fraudsters may target customers by their age
  - fraudsters may use specific groups of transaction types

Coding: Lookup and Fraud

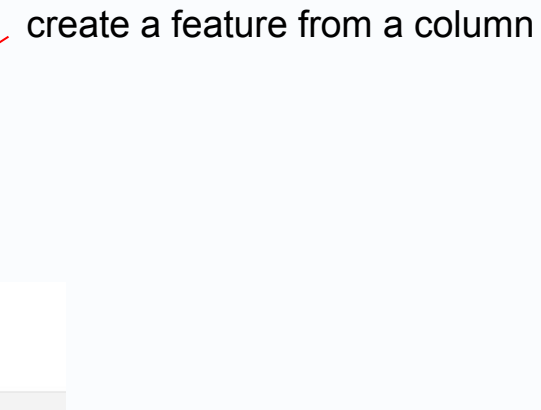
What age is the customer?

```
# convert date of birth to a feature
date_of_birth = bank_customer_view.DateOfBirth.as_feature("date of birth")

# calculate the customer age in three steps: year, month, day of month
customer_age = RequestColumn.point_in_time().dt.year - date_of_birth.dt.year

filter = RequestColumn.point_in_time().dt.month < date_of_birth.dt.month
customer_age[filter] = customer_age[filter] - 1

filter = (RequestColumn.point_in_time().dt.month == date_of_birth.dt.month) & \
    (RequestColumn.point_in_time().dt.day < date_of_birth.dt.day)
customer_age[filter] = customer_age[filter] - 1

customer_age.name = 'CustomerAge'
```

age will be a transformation of a feature

the point in time at which the feature is evaluated

Coding: Lookup and Fraud



What transaction group does the transaction belong to?

```
# create a feature from the TransactionGroup column
transaction_group = card_transactions_view.TransactionGroup.as_feature("transaction group")
```

Statistic Signals

An aggregation that is an undefined combination of signal types

Metrics

- Maximum or maximum attribute of events occurring over a time window.
- Aggregation as at a point in time

Examples

- What was the maximum temperature over the past 24 hours?
- How many bank accounts does a customer have?

Ideation: Statistic and Fraud

- Entities:
  - customer
  - credit card
- Attributes:
  - count
- Ideas:
  - the more credit cards a customer has, the more likely that the details of one of those cards will be stolen

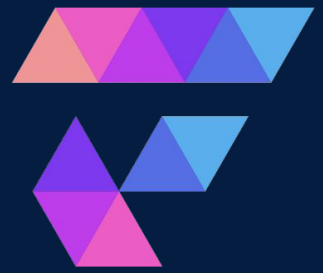
Coding: Statistic and Fraud

How many active credit cards does the customer have?

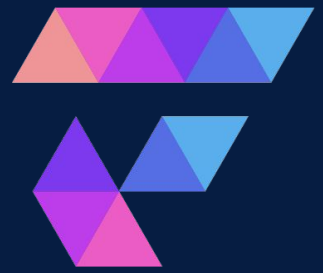
```
# filter the credit card table to only include active cards
active_credit_cards = credit_card_view[credit_card_view.closed_at.isnull()]

# count the number of active credit cards for each customer
customer_active_credit_card_count = active_credit_cards.groupby(
    "BankCustomerID"
).aggregate_asat(
    None,
    method=fb.AggFunc.COUNT,
    feature_name="customer active credit card count",
)
```

slowly changing dimension data so no time window

use aggregate_as_at to aggregate slow changing dimension data at a single point in time



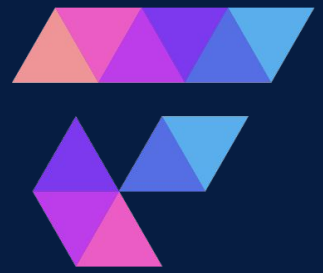
Q&A



Module 3

Module 3: Goals

- Combine features into coherent feature lists
- Check the primary entities of features and feature lists



Feature Lists

Coding: Feature List

Combine All Signal Types into a Feature List

```
diverse_feature_list = fb.FeatureList([
    recency_features,
    cross_aggregation_features,
    similarity_features,
    stability_features,
    clumpiness_features,
    diversity_features,
    change_features,
    timing_features,
    location_features,
    frequency_features,
    monetary_features,
    lookup_features,
    statistic_features,
], name="diverse feature list for fraud detection")
diverse_feature_list.save()
```

create a feature list

saving a feature list also causes its features to be saved

Feature List

```
diverse_feature_list.list_features()
```

27.1s Python

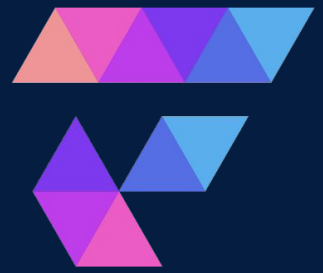
	id	name	version	dtype	readiness	online_enabled	tables	primary_tables	entities	primary_entities	created_at
0	647889c6fdb8f65d424afe3e	customer active credit card count	V230601	FLOAT	DRAFT	False	[CREDITCARD]	[CREDITCARD]	[bank_customer]	[bank_customer]	2023-06-01 12:09:36.479
1	647889b7fdb8f65d424afe3c	transaction group	V230601	VARCHAR	DRAFT	False	[CARDTRANSACTIONS, CARDTRANSACTIONGROUPS]	[CARDTRANSACTIONS]	[card_transaction]	[card_transaction]	2023-06-01 12:09:35.166
2	647889b3fdb8f65d424afe24	CustomerAge	V230601	INT	DRAFT	False	[BANKCUSTOMER]	[BANKCUSTOMER]	[bank_customer]	[bank_customer]	2023-06-01 12:09:32.380
3	6478899cfdb8f65d424afe1e	mean fraud transaction amount 4w	V230601	FLOAT	DRAFT	False	[CREDITCARD, CARDTRANSACTIONS, CARDFRAUDSTATUS]	[CARDFRAUDSTATUS]	[bank_customer]	[bank_customer]	2023-06-01 12:09:25.399
4	6478899bfdb8f65d424afe1c	customer interest charges 30d	V230601	FLOAT	DRAFT	False	[CREDITCARD, CARDTRANSACTIONS, CARDTRANSACTION...	[CARDTRANSACTIONS]	[bank_customer]	[bank_customer]	2023-06-01 12:09:18.326
5	64788999fdb8f65d424afe1a	Customer Average Weekly Purchases Total 8w	V230601	FLOAT	DRAFT	False	[CREDITCARD, CARDTRANSACTIONS, CARDTRANSACTION...	[CARDTRANSACTIONS]	[bank_customer]	[bank_customer]	2023-06-01 12:09:10.798
6	64788985fdb8f65d424afe16	Customer Average Weekly Transaction Count 8w	V230601	FLOAT	DRAFT	False	[CREDITCARD, CARDTRANSACTIONS]	[CARDTRANSACTIONS]	[bank_customer]	[bank_customer]	2023-06-01 12:09:07.959
7	6478897ffdb8f65d424afe0e	fraudulent transactions 24h	V230601	FLOAT	DRAFT	False	[CARDFRAUDSTATUS]	[CARDFRAUDSTATUS]	[]	[]	2023-06-01 12:09:05.171
8	6478895dfdb8f65d424afe0c	Customer State Distance	V230601	FLOAT	DRAFT	False	[BANKCUSTOMER, STATEDETAILS]	[BANKCUSTOMER, STATEDETAILS]	[bank_customer, USA_state]	[bank_customer]	2023-06-01 12:08:39.643
9	6478890ffdb8f65d424afd4e	customer longitude	V230601	FLOAT	DRAFT	False	[BANKCUSTOMER]	[BANKCUSTOMER]	[bank_customer]	[bank_customer]	2023-06-01 12:08:20.853
10	6478890efdb8f65d424afd4c	customer latitude	V230601	FLOAT	DRAFT	False	[BANKCUSTOMER]	[BANKCUSTOMER]	[bank_customer]	[bank_customer]	2023-06-01 12:08:19.584
11	647888effdb8f65d424afd4a	Customer Transaction Day of Week Entropy 90d	V230601	FLOAT	DRAFT	False	[CREDITCARD, CARDTRANSACTIONS, CARDTRANSACTION...	[CARDTRANSACTIONS]	[bank_customer]	[bank_customer]	2023-06-01 12:08:17.945
12	647888ebfdb8f65d424afd46	customer transaction hour frequency 90d	V230601	FLOAT	DRAFT	False	[CREDITCARD, CARDTRANSACTIONS, CARDTRANSACTION...	[CARDTRANSACTIONS]	[bank_customer, card_transaction]	[card_transaction]	2023-06-01 12:08:14.815
13	647888e8fdb8f65d424afd42	transaction hour	V230601	INT	DRAFT	False	[CARDTRANSACTIONS, CARDTRANSACTIONGROUPS]	[CARDTRANSACTIONS]	[card_transaction]	[card_transaction]	2023-06-01 12:08:09.913
14	647888d6fdb8f65d424afd3e	card issued 90d	V230601	FLOAT	DRAFT	False	[CREDITCARD]	[CREDITCARD]	[bank_customer]	[bank_customer]	2023-06-01 12:08:06.851
15	647888d5fdb8f65d424afd3a	customer address changes 365d	V230601	FLOAT	DRAFT	False	[BANKCUSTOMER]	[BANKCUSTOMER]	[bank_customer]	[bank_customer]	2023-06-01 12:08:01.498
16	647888bdfdb8f65d424afd38	Customer Transaction Description Diversity 28d	V230601	FLOAT	DRAFT	False	[CREDITCARD, CARDTRANSACTIONS]	[CARDTRANSACTIONS]	[bank_customer]	[bank_customer]	2023-06-01 12:07:58.794
17	647888bbfdb8f65d424afd34	Customer Transaction Amount Diversity 28d	V230601	FLOAT	DRAFT	False	[CREDITCARD, CARDTRANSACTIONS]	[CARDTRANSACTIONS]	[bank_customer]	[bank_customer]	2023-06-01 12:07:54.397
18	6478889ffdb8f65d424afd2c	Card Transactions Timing Coefficient of Variat...	V230601	FLOAT	DRAFT	False	[CREDITCARD, CARDTRANSACTIONS]	[CARDTRANSACTIONS]	[bank_customer]	[bank_customer]	2023-06-01 12:07:48.495
19	64788898fdb8f65d424afd26	Card Transactions Timing Clumpiness Entropy (z...	V230601	FLOAT	DRAFT	False	[CREDITCARD, CARDTRANSACTIONS]	[CARDTRANSACTIONS]	[bank_customer]	[bank_customer]	2023-06-01 12:07:35.032
20	64788854fdb8f65d424afd10	Customer Transaction Desc Stability 1w1yr	V230601	FLOAT	DRAFT	False	[CREDITCARD, CARDTRANSACTIONS]	[CARDTRANSACTIONS]	[bank_customer]	[bank_customer]	2023-06-01 12:07:24.955
21	64788852fdb8f65d424afd0a	Transaction Z Score 365d	V230601	FLOAT	DRAFT	False	[CREDITCARD, CARDTRANSACTIONS]	[CARDTRANSACTIONS]	[bank_customer, card_transaction]	[card_transaction]	2023-06-01 12:07:20.830
22	64788832fdb8f65d424afcfd	Customer State Similarity 28d	V230601	FLOAT	DRAFT	False	[BANKCUSTOMER, CREDITCARD, CARDTRANSACTIONS]	[CARDTRANSACTIONS]	[bank_customer, USA_state]	[bank_customer]	2023-06-01 12:07:15.766
23	64788826fdb8f65d424afcf4	Z-Score of Transaction Amount vs Recent Fraudu...	V230601	FLOAT	DRAFT	False	[CARDTRANSACTIONS, CARDFRAUDSTATUS]	[CARDTRANSACTIONS, CARDFRAUDSTATUS]	[card_transaction]	[card_transaction]	2023-06-01 12:07:07.864
24	6478881efdb8f65d424afce6	Matches Latest Fraudulent Transaction Description	V230601	VARCHAR	DRAFT	False	[CARDTRANSACTIONS]	[CARDTRANSACTIONS]	[card_transaction]	[card_transaction]	2023-06-01 12:07:02.580
25	6478881bfdb8f65d424afce2	Matches Recent Fraudulent Transaction Description	V230601	FLOAT	DRAFT	False	[CARDTRANSACTIONS, CARDFRAUDSTATUS]	[CARDTRANSACTIONS, CARDFRAUDSTATUS]	[card_transaction]	[card_transaction]	2023-06-01 12:06:58.881
26	6478880bfdb8f65d424afcda	Fraudulent Most Frequent Category	V230601	VARCHAR	DRAFT	False	[CARDTRANSACTIONS, CARDFRAUDSTATUS]	[CARDFRAUDSTATUS]	[]	[]	2023-06-01 12:06:54.747
27	64788808fdb8f65d424afcd6	Customer Highest Spend Category	V230601	VARCHAR	DRAFT	False	[CREDITCARD, CARDTRANSACTIONS]	[CARDTRANSACTIONS]	[bank_customer]	[bank_customer]	2023-06-01 12:06:49.643
28	647887fbfdb8f65d424afcd0	Latest Fraud Description	V230601	VARCHAR	DRAFT	False	[CARDTRANSACTIONS, CARDFRAUDSTATUS]	[CARDFRAUDSTATUS]	[]	[]	2023-06-01 12:06:46.365
29	647887f9fdb8f65d424afcce	Days Since Customer Latest Fraud	V230601	FLOAT	DRAFT	False	[CREDITCARD, CARDTRANSACTIONS, CARDFRAUDSTATUS]	[CARDFRAUDSTATUS]	[bank_customer]	[bank_customer]	2023-06-01 12:06:40.164

Feature List Primary Entity

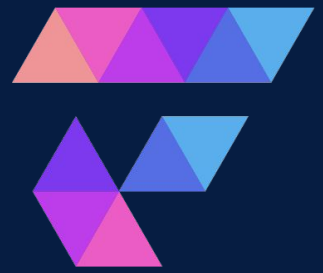
```
# show the primary entity of the feature list
diverse_feature_list.info()['primary_entity']
```

✓ 0.5s

```
[{'name': 'card_transaction',
  'serving_names': ['CARDTRANSACTIONID'],
  'catalog_name': 'credit card playground 20230601:1955'}]
```



Q&A



Conclusion

Conclusion

Use signal types and entities for inspiration and information.

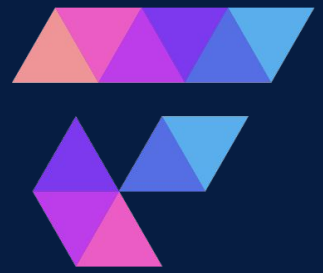
In this workshop we quickly created 30 diverse features spanning 13 signal types and 3 primary entities.

Inspiration

- Lists of signal types and entities can prompt and remind you to engineer missing features

Information

- Features with different signal types tend to be less correlated
- Business SMEs can intuitively understand signal types



Thank You!