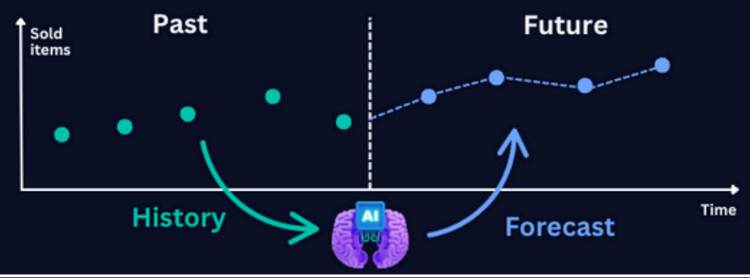
ML System Modeling and Design

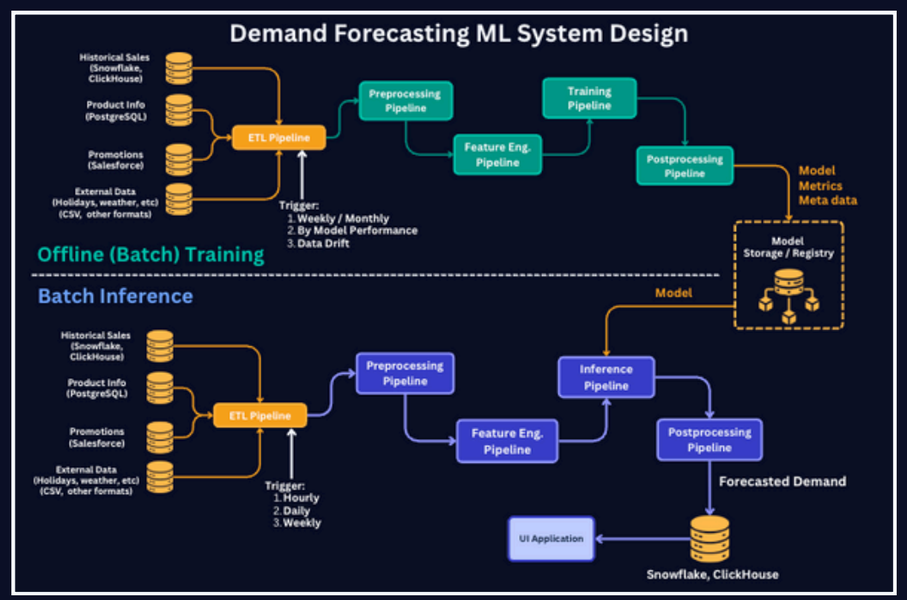
Business Problem

Demand forecasting is about predicting how much of something people will need in the future, like products, staff, or delivery capacity.

Here are some typical problems it helps solve:

- **How many items will we sell? → So we don't run out or overstock.**
- **How much electricity will we need to generate? → So we can optimize prices.**
- **How much should we produce? → To avoid under- or over-manufacturing.**
- **How many trucks or deliveries will we need? → So logistics teams can plan ahead.**





Training

```
[38]  df.reset_index(drop=True, inplace=True)
```

```
[39]  x_train, x_test, y_train, y_test = prepare_dataset(df, train_fraction=0.8)
```

```
[40]  x_train
```

	season	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	...	weekday
0	1	1	0	0	6	0	1	0.24	0.2879	0.81	...	
1	1	1	1	0	6	0	1	0.22	0.2727	0.80	...	
2	1	1	2	0	6	0	1	0.22	0.2727	0.80	...	
3	1	1	3	0	6	0	1	0.24	0.2879	0.75	...	
4	1	1	4	0	6	0	1	0.24	0.2879	0.75	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
13898	3	8	7	0	2	1	2	0.70	0.6667	0.74	...	
13899	3	8	8	0	2	1	2	0.70	0.6515	0.70	...	
13900	3	8	9	0	2	1	2	0.70	0.6667	0.74	...	
13901	3	8	10	0	2	1	2	0.74	0.6970	0.70	...	
13902	3	8	11	0	2	1	2	0.76	0.7273	0.66	...	

13903 rows × 31 columns

```
# Step 1: Define the Optuna objective with TimeSeriesSplit on x_train
def objective(trial):
    np.random.seed(42)
    params = {
        "learning_rate": trial.suggest_float("learning_rate", 0.01, 0.2),
        "depth": trial.suggest_int("depth", 3, 8),
        "l2_leaf_reg": trial.suggest_float("l2_leaf_reg", 0.5, 5),
        "iterations": 1000,
        "loss_function": "RMSE",
        "verbose": 0
    }
```

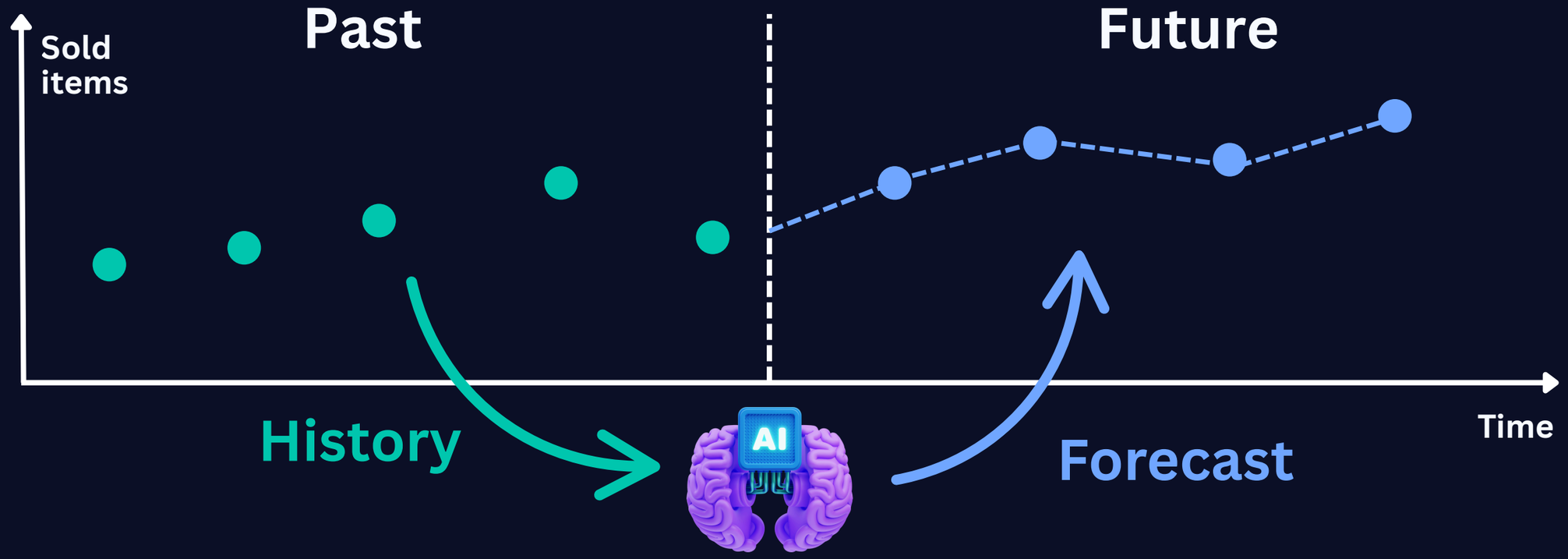
Demand Forecasting Modeling and ML System Design

Business Problem

Demand forecasting is about predicting how much of something people will need in the future, like products, staff, or delivery capacity.

Here are some typical problems it helps solve:

- **How many items will we sell? → So we don’t run out or overstock.**
- **How much electricity will we need to generate? → So we can optimize prices.**
- **How much should we produce? → To avoid under- or over-manufacturing.**
- **How many trucks or deliveries will we need? → So logistics teams can plan ahead.**



Typical Features:

- Lag features: Sales over the past 7, 14, 30 days
- Rolling statistical (means/medians/std) (e.g., 7-day avg sales)
- Price elasticity: sales vs price changes
- Promotion indicators (is_promo_active)
- Time-based flags (weekend, holiday, season)
- Weather impact indicators (temperature bin, rain flag)

Typical Target:

- Number of sold units in the next X days (sum / mean / multistep values per day)
- Forecasted demand (sum / mean / multistep values per day)
- Forecasted revenue (sum / mean / multistep values per day)

Typical Models:

- Gradient Boosting (XGBoost, Catboost, LightGBM)
- Prophet or NeuralProphet to easily account for trend + seasonality
- DeepAR, Temporal Fusion Transformer (TFT)
- ARIMA/SARIMA/Holt-Winters for univariate classical forecasting

Demand Forecasting ML System Design

Historical Sales (Snowflake, ClickHouse)

Product Info (PostgreSQL)

Promotions (Salesforce)

External Data (Holidays, weather, etc) (CSV, other formats)

ETL Pipeline

Preprocessing Pipeline

Feature Eng. Pipeline

Training Pipeline

Postprocessing Pipeline

PREFECT

METAFLOW

kedro

Model
Metrics
Meta data

Model Storage / Registry

mlflow

comet

Trigger:
1. Weekly / Monthly
2. By Model Performance
3. Data Drift

Offline (Batch) Training

Batch Inference

Historical Sales (Snowflake, ClickHouse)

Product Info (PostgreSQL)

Promotions (Salesforce)

External Data (Holidays, weather, etc) (CSV, other formats)

ETL Pipeline

Preprocessing Pipeline

Feature Eng. Pipeline

Model

Inference Pipeline

Postprocessing Pipeline

Forecasted Demand

Trigger:
1. Hourly
2. Daily
3. Weekly

plotly | Dash

Streamlit

UI Application

Snowflake, ClickHouse

Churn Prediction Modeling and ML System Design

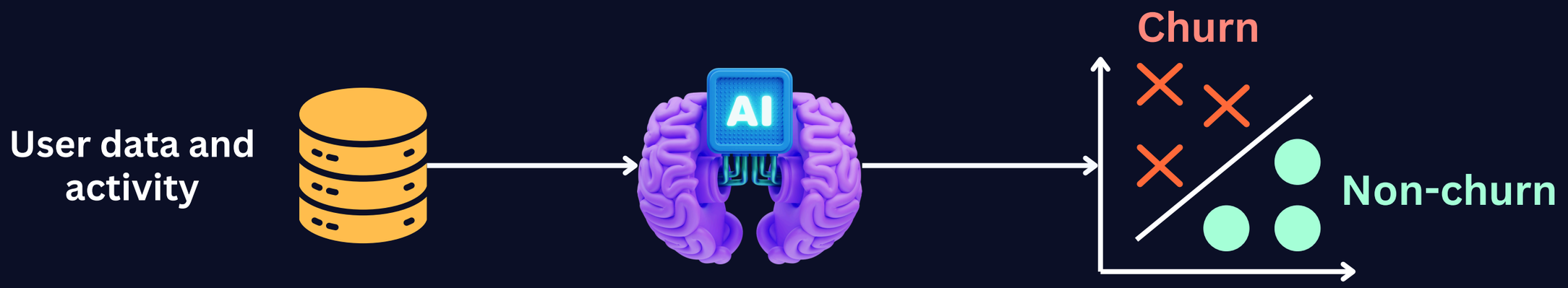
Business Problem

Churn prediction systems identify users or customers likely to cancel a subscription, stop purchasing, or disengage from a service.

Common business goals include:

- **Improving customer retention by identifying high-risk users early.**
- **Reducing churn-driven revenue loss.**
- **Supporting targeted interventions such as discounts, reactivation offers, or proactive outreach.**

Churn prediction can be modeled as binary classification (will churn / will not churn) or as risk scoring (likelihood to churn).



Churn Prediction ML System Design

Typical Features:

- **Recency**: Days since last login or activity
- **Frequency**: Number of logins, sessions, or purchases in the last X days
- **Engagement** duration: Average session length over the last week or month
- **Feature usage counts**: How often key features were used (e.g., reports generated, messages sent)
- **Plan or tier info**: Subscription type, feature access
- **Support activity**: Number of support tickets filed, time to resolution
- **Billing patterns:** Failed payments, payment method changes, recent upgrades/downgrades
- **Marketing interaction:** Click-through rate on emails, offers redeemed
- **Net promoter score (NPS)** or survey feedback when available
- **Account age:** Days since sign-up or subscription
- **Inactivity streak:** Longest stretch of inactivity in recent time window

Typical Target:

- Churn / non-churn label within churn window (e.g. next 14 days)

Typical Models:

- Random Forest and Gradient Boosting (XGBoost, LightGBM)
- Logistic Regression for interpretable binary classification
- Neural Networks for behavioral and event sequence modeling
- Autoencoders or Isolation Forests for anomaly-based churn risk

Churn Prediction ML System Design

CRM (PostgreSQL)

Usage logs (Snowflake, ClickHouse)

Billing Data (Saas Platforms)

Support tickets (Saas Platforms)

ETL Pipeline

Preprocessing Pipeline

Feature Eng. Pipeline

Training Pipeline

Postprocessing Pipeline

PREFECT

METAFLOW

kedro

Trigger:
1. Weekly / Monthly
2. By Model Performance
3. Data Drift

Model
Metrics
Meta data

Model Storage / Registry

mlflow

comet

Offline (Batch) Training

Batch Inference

(Usually) Daily Trigger

CRM (PostgreSQL)

Usage logs (Snowflake)

Billing Data (Saas Platforms)

Support tickets (Saas Platforms)

ETL Pipeline

Preprocessing Pipeline

Feature Eng. Pipeline

Inference Pipeline

Postprocessing Pipeline

Model

Churn / Non-churn

customer_id: churn_risk (PostgreSQL)

Alert about high churn customer

Personalized Outreach Action

Fraud Detection Modeling and ML System Design

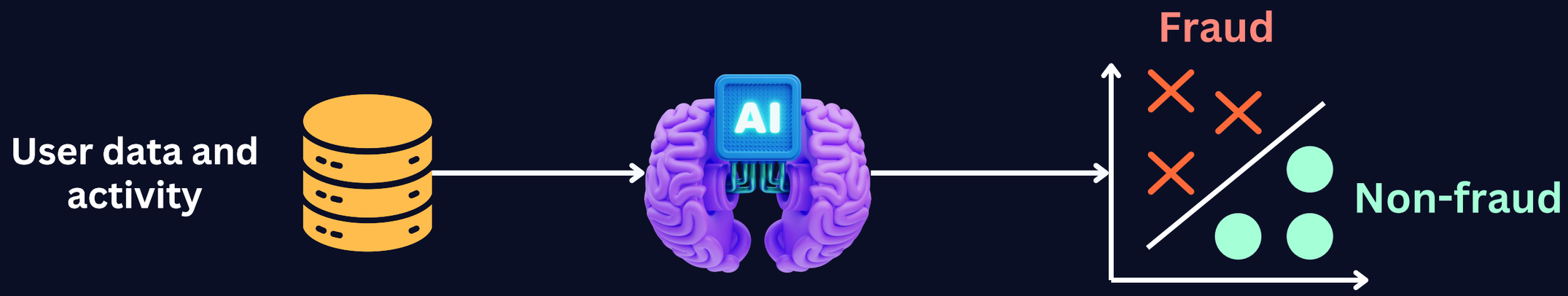
Business Problem

Fraud detection systems aim to identify and prevent fraudulent activity in real time or near real time, reducing financial losses, minimizing reputational damage, and ensuring compliance with regulations.

Common use cases include:

- **Detecting unauthorized or suspicious transactions.**
- **Identifying account takeovers or identity theft.**
- **Flagging new account fraud.**
- **Monitoring unusual patterns in user behavior or transactions.**

These systems are often used to trigger alerts, hold transactions, or escalate to human review.



Fraud Detection ML System Design

Typical Features:

Transactional Patterns:

- Amount, frequency, and recency of transactions
- Velocity: spend rate or number of actions in short windows (e.g., 10 mins)

Device & Location Signals:

- New IP, device, or location
- Distance between recent locations

Behavioral Signals:

- Time-of-day anomalies
- Login success/failure ratios

Historical & Graph Features:

- Past fraud flags
- Account age
- Shared devices, cards, or addresses

Typical Target:

- Fraud / non-fraud at the transaction time (real-time)

Typical Models:

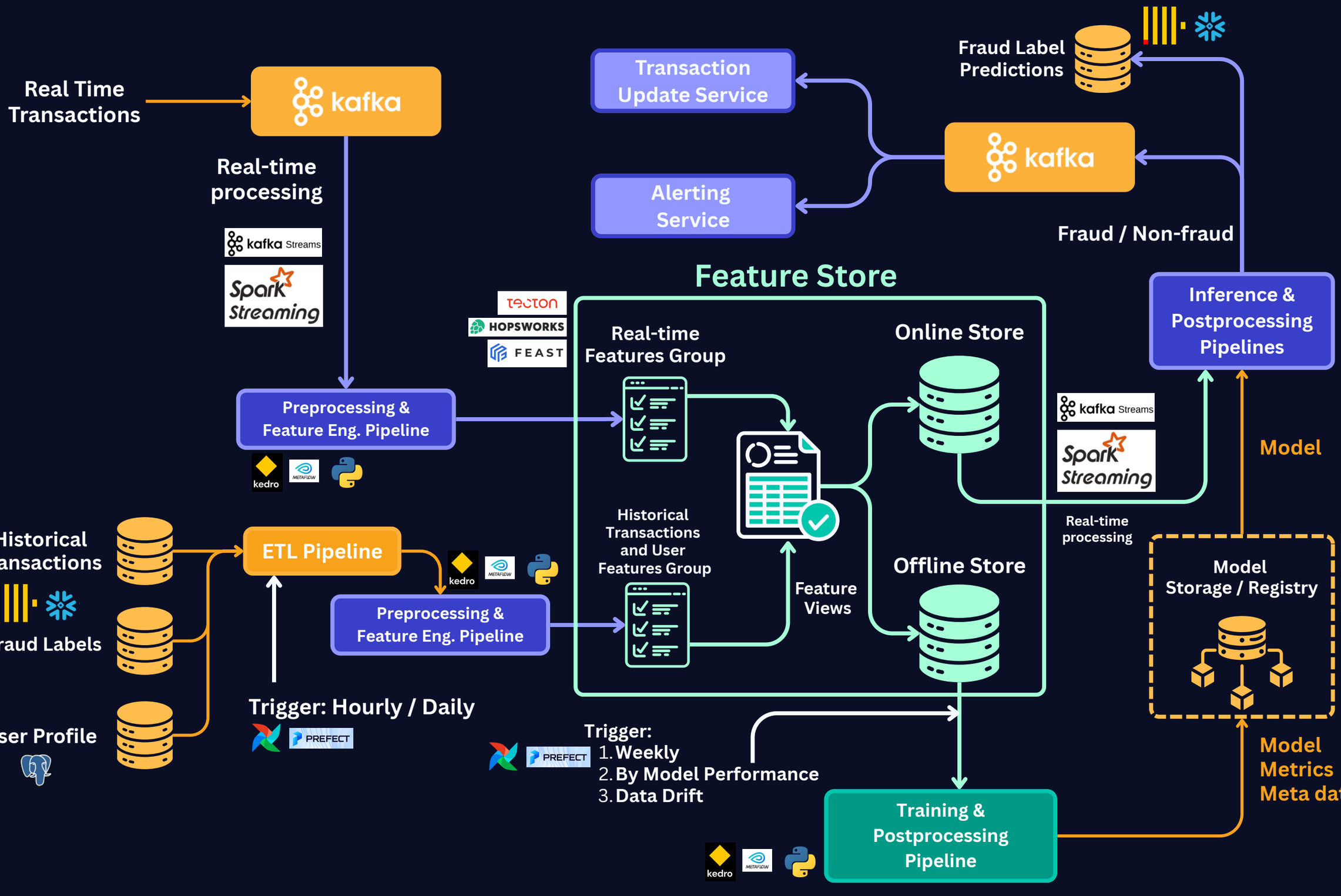
Supervised Classification:

- Logistic Regression
- Random Forest
- XGBoost / LightGBM

Unsupervised Anomaly Detection:

- Isolation Forest
- Autoencoders

Fraud Detection ML System Design



Recommendation System Modeling and ML System Design

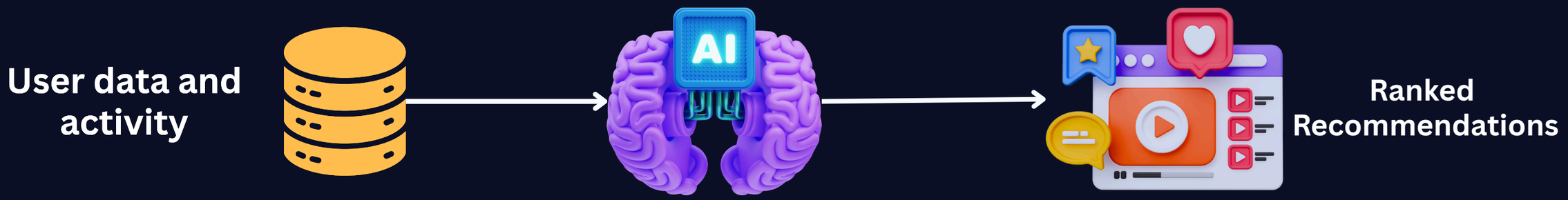
Business Problem

Video recommendation systems aim to deliver personalized content suggestions to maximize user engagement, watch time, and satisfaction.

Key business goals include:

- **Increasing total watch time per user session.**
- **Enhancing user retention and return visits.**
- **Promoting high-value or trending content.**
- **Reducing churn by keeping users engaged.**

These systems address questions like: "What should we show next to this user to keep them watching/buying?"



Movie / Video Recommendation System ML System Design

Typical Features:

Basic:

- User ID (always included as a base embedding)
- Location (city, country, timezone)
- Language preference
- Signup date/tenure (how long they've been using the service)

Behavioral:

- Recent watch history (movie IDs, genres, last N watched items)
- Aggregated preferences (average genre distribution, avg rating given)
- Time-of-day activity (morning/night watcher)
- Frequency (daily vs. occasional usage)

Demographic (if available & privacy-compliant):

- Age group
- Gender
- Subscription tier (free, premium)

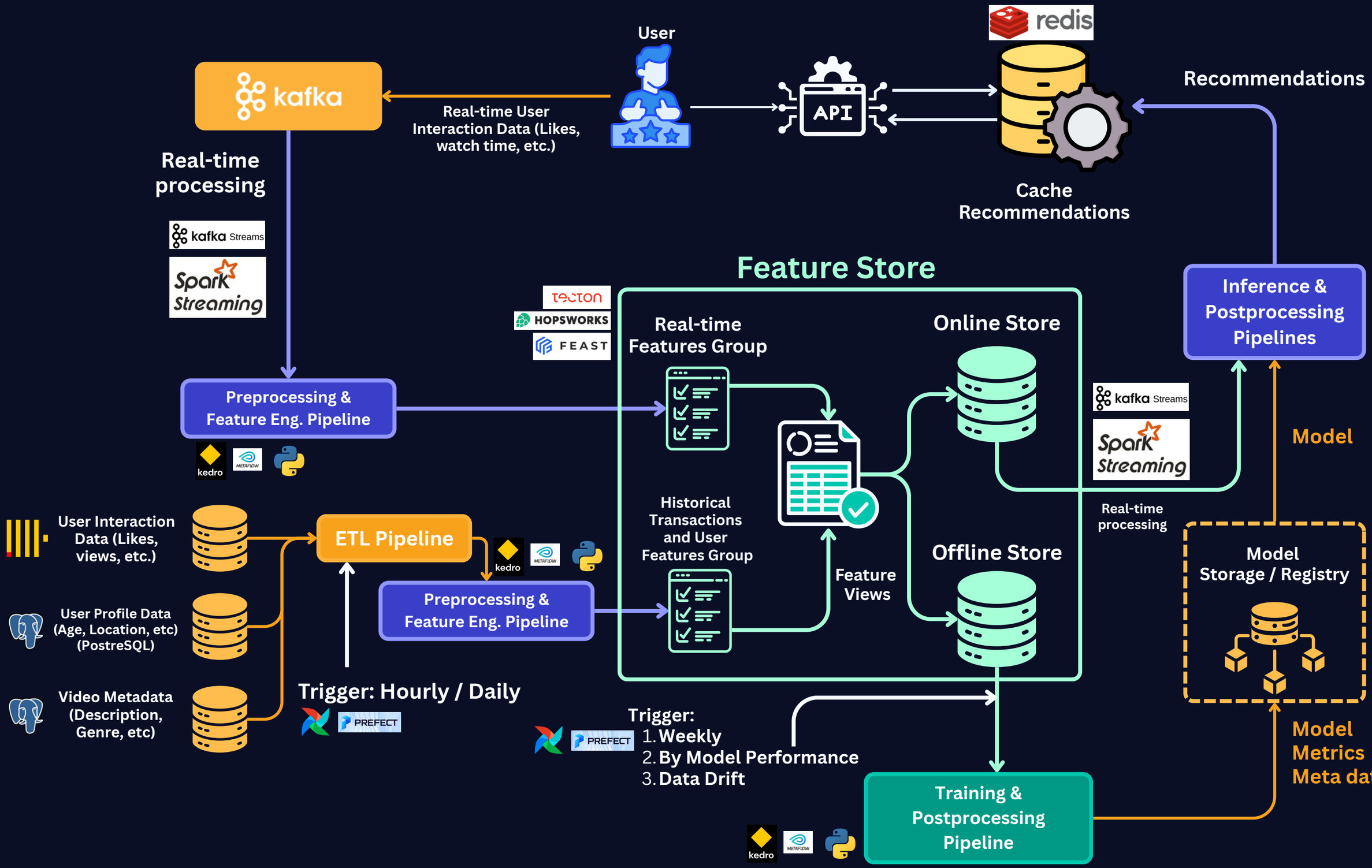
Typical Target:

- ranking value for a movie

Typical Models:

- Matrix Factorization
- Two-tower neural networks
- Hybrid Models

Movie / Video Recommendation System ML System Design



Matrix Factorization Model

	m_1	m_2	m_3	m_4
latent_1	2.058	2.026	1.951	1.594
latent_2	-0.206	0.263	-0.982	1.134

Movies

Users

	m_1	m_2	m_3	m_4
u_1	5	?	3	?
u_2	?	4	?	4
u_3	?	?	5	2

≈

	latent_1	latent_2
u_1	2.272	1.086
u_2	2.15	-0.28
u_3	2.214	-1.09

	m_1	m_2	m_3	m_4
u_1	4.45	4.89	3.38	4.85
u_2	4.43	4.34	4.22	3.40
u_3	4.78	4.20	5.39	2.30

$$R \approx U \times M^T \approx \hat{R}$$

Observed loss

$$\text{loss} = \sum_{(i,j) \in \text{obs}} \left(r_{ij} - u_i m_j \right)^2$$

$u_i m_j \rightarrow \hat{r}_{ij}$

Movies

Users

	m_1	m_2	m_3	m_4
u_1	5	?	3	?
u_2	?	4	?	4
u_3	?	?	5	2

Combined loss

$$\text{loss} = \sum_{(i,j) \in \text{obs}} \left(r_{ij} - u_i m_j \right)^2 + W \sum_{(i,j) \notin \text{obs}} \left(r_{ij} - u_i m_j \right)^2$$

$u_i m_j \rightarrow \hat{r}_{ij}$

Movies

Users

	m_1	m_2	m_3	m_4
u_1	5	0	3	0
u_2	0	4	0	4
u_3	0	0	5	2