ML Feature DSL Proposal

Overview

The ML Feature DSL is a domain-specific language that enables clear, reusable, and consistent definition of features used in machine learning models. It abstracts away complex SQL or Python transformation code, allowing teams to define feature logic once and generate compatible code across training, serving, monitoring, and reporting environments.

Goals

- Define machine learning features in a human-readable, declarative format
- Generate code for multiple targets (Python, SQL, API schemas, monitoring)
- Eliminate code duplication and inconsistency across teams
- Make feature logic auditable and shareable

DSL Syntax (Input)

Example:

```
feature: is_active = days_since_last_login < 30

feature: low_spender = total_spent < 100

feature: no_recent_support = support_tickets_last_90d == 0
```

Each feature: line declares a new machine learning feature. The syntax supports simple expressions with logical, comparison, and function operations.

Outputs Generated Automatically

1. Python (for training in Pandas)

```
df["is_active"] = df["days_since_last_login"] < 30
df["low_spender"] = df["total_spent"] < 100
df["no_recent_support"] = df["support_tickets_last_90d"] == 0
```

2. SQL (for data extraction or dashboards)

```
CASE WHEN days_since_last_login < 30 THEN TRUE ELSE FALSE END AS is_active,
total_spent < 100 AS low_spender,
support_tickets_last_90d = 0 AS no_recent_support
```

3. Monitoring Config (e.g. YAML)

```
features:
 - name: is_active
   drift_check: ks_test
   threshold: 0.05
 - name: low_spender
   drift_check: js_divergence
   threshold: 0.1
```

4. API Schema (JSON Input for Real-Time Prediction)

```
{
 "input": {
   "days_since_last_login": "number",
   "total_spent": "number",
   "support_tickets_last_90d": "number"
 },
 "features": ["is_active", "low_spender", "no_recent_support"]
}
```

Supported DSL Input Types

Type	Example Syntax	Description
Comparison	`x > 5`, `y == 0`	Boolean conditions
Logical	`a AND b`, `x OR y`	Combine multiple conditions
Function	`hour(time) > 20`, `bucket(age, [18,30,50])`	Feature transformation functions
Arithmetic	`total / count > 5`	Derived numeric features

These inputs are parsed and transformed into executable code depending on the target environment.

Benefits

- **Single Source of Truth**: One DSL file generates all environments' code.
- **Team Collaboration**: Data scientists, engineers, and analysts share the same logic.
- **Auditability**: Easy to track, version, and explain.
- **Consistency**: Same features in training, production, and analytics.
- **Productivity**: Build new models faster with reusable feature libraries.

Use Cases

Churn Prediction

```
feature: is_inactive = days_since_last_login > 45

feature: low_engagement = weekly_sessions < 2
```

Fraud Detection

```
feature: large_txn = transaction_amount > 1000

feature: night_txn = hour(transaction_time) > 22
```

Recommendation Engine

```
feature: viewed_recently = days_since_last_view < 7

feature: repeat_buyer = total_orders > 2
```

Conclusion

This proposal introduces an ML Feature DSL that enables clean, reusable, and consistent feature engineering at scale. The simplicity of the syntax hides its powerful ability to unify ML workflows across environments. Adoption of this DSL can significantly reduce duplication, bugs, and inconsistencies in real-world ML projects.

DSL Example – Customer Lifetime Value Prediction

```
# User Activity
feature: is_recent_login = days_since_last_login < 14
feature: active_last_month = monthly_sessions_last_30d > 5
feature: multi_device_user = device_count > 1

# Purchase Behavior
feature: avg_order_value = total_spent / total_orders
feature: high_spender = total_spent > 1000
feature: frequent_buyer = total_orders > 10
feature: last_purchase_recent = days_since_last_purchase < 30

# Subscription & Retention
feature: subscription_active = subscription_status == "active"
feature: churn_risk = days_since_subscription_renewal > 45

# Support Interactions
feature: frequent_complainer = support_tickets_last_90d > 3
feature: unresolved_issues = open_support_tickets > 0

# Engagement
feature: viewed_products_last_week = product_views_last_7d > 10
feature: marketing_engaged = marketing_email_clicks_last_30d > 3

```

Python Output (Pandas)

```
df["is_recent_login"] = df["days_since_last_login"] < 14
df["active_last_month"] = df["monthly_sessions_last_30d"] > 5
df["multi_device_user"] = df["device_count"] > 1

df["avg_order_value"] = df["total_spent"] / df["total_orders"]
df["high_spender"] = df["total_spent"] > 1000
df["frequent_buyer"] = df["total_orders"] > 10
df["last_purchase_recent"] = df["days_since_last_purchase"] < 30

df["subscription_active"] = df["subscription_status"] == "active"
df["churn_risk"] = df["days_since_subscription_renewal"] > 45

df["frequent_complainer"] = df["support_tickets_last_90d"] > 3
df["unresolved_issues"] = df["open_support_tickets"] > 0

df["viewed_products_last_week"] = df["product_views_last_7d"] > 10
df["marketing_engaged"] = df["marketing_email_clicks_last_30d"] > 3

```

SQL Output

```sql
days_since_last_login < 14 AS is_recent_login,
monthly_sessions_last_30d > 5 AS active_last_month,
device_count > 1 AS multi_device_user,

total_spent / NULLIF(total_orders, 0) AS avg_order_value,
total_spent > 1000 AS high_spender,
total_orders > 10 AS frequent_buyer,
days_since_last_purchase < 30 AS last_purchase_recent,

subscription_status = 'active' AS subscription_active,
days_since_subscription_renewal > 45 AS churn_risk,

support_tickets_last_90d > 3 AS frequent_complainer,
open_support_tickets > 0 AS unresolved_issues,

product_views_last_7d > 10 AS viewed_products_last_week,
marketing_email_clicks_last_30d > 3 AS marketing_engaged

```

Monitoring Config (YAML)

```yaml
features:
  - name: is_recent_login
    drift_check: ks_test
    threshold: 0.05
  - name: avg_order_value
    drift_check: wasserstein_distance
    threshold: 10
  - name: churn_risk
    drift_check: ks_test
    threshold: 0.05
  - name: marketing_engaged
    drift_check: js_divergence
    threshold: 0.1

```

API Input Schema (JSON)

```json
{
  "input": {
    "days_since_last_login": "number",
    "monthly_sessions_last_30d": "number",
    "device_count": "number",
    "total_spent": "number",
    "total_orders": "number",
    "days_since_last_purchase": "number",
    "subscription_status": "string",
    "days_since_subscription_renewal": "number",
    "support_tickets_last_90d": "number",
    "open_support_tickets": "number",
    "product_views_last_7d": "number",
    "marketing_email_clicks_last_30d": "number"
  },
  "features": [
    "is_recent_login",
    "active_last_month",
    "multi_device_user",
    "avg_order_value",
    "high_spender",
    "frequent_buyer",
    "last_purchase_recent",
    "subscription_active",
    "churn_risk",
    "frequent_complainer",
    "unresolved_issues",
    "viewed_products_last_week",
    "marketing_engaged"
  ]
}

```