03_feature_engineering

December 8, 2025

1 Feature Engineering

This notebook creates all features needed for the analysis: - Exposure counting features (critical for RQ1) - Temporal features - Campaign-level aggregated features (for RQ2) - User-level features - Context features

---

1.1 Prerequisites

Input Data: `../data/processed/data_with_exposures.csv` (created by `02_exploratory_analysis.ipynb`)

Output Data: `../data/processed/data_with_all_features.csv` (used by notebooks 04, 05, 06)

1.2 Data Pipeline Position

`01_data_acquisition → 02_exploratory_analysis → [03_feature_engineering.ipynb] → 04-06 Analysis`

1.3 Source Module Dependencies

This notebook calls functions from the `src/` directory:

Module	Function	Description
`src.feature_engineering`	`create_all_features()`	Main function that computes all features (exposure counts, temporal, campaign-level, user-level)
`src.data_loader`	`load_config()`	Loads configuration from `config/config.yaml`

1.3.1 `create_all_features()` Details

Location: `src/feature_engineering.py`

Inputs: - `df`: DataFrame with columns `uid`, `campaign`, `timestamp`, `click` - `user_col`, `campaign_col`, `time_col`, `click_col`: Column name mappings - `windows_hours`: List of time windows for exposure counting (from config)

Outputs: - DataFrame with 40+ new feature columns including: - `exposure_count`: Sequential exposure number (1st, 2nd, 3rd impression) - `exposure_count_24h`,

`exposure_count_168h`, `exposure_count_720h`: Time-windowed exposure counts - `hours_since_first_exposure`, `hours_since_last_exposure`: Recency features - `campaign_overall_ctr`, `campaign_total_impressions`: Campaign-level aggregations - `user_overall_ctr`, `user_total_clicks`: User-level aggregations - `hour_of_day`, `day_of_week`, cyclical encodings: Temporal features

[4]:
```
import sys
sys.path.append('../')

import pandas as pd
import numpy as np
from src.feature_engineering import create_all_features
from src.data_loader import load_config

# Load configuration
config = load_config('../config/config.yaml')

# Load data (from previous notebook)
df = pd.read_csv('../data/processed/data_with_exposures.csv')
print(f"Loaded {len(df):,} records")
```

```
Loaded 500,000 records
```

1.4 Step 1: Create All Features

The `create_all_features()` function from `src/feature_engineering.py` computes all features in a single pass. This function:

1. **Exposure counts** (total and time-windowed: 24h, 168h, 720h)
2. **Exposure recency features** (hours since first/last exposure)
3. **Exposure intensity features** (exposures per day, average gap)
4. **Temporal features** (hour, day of week, with cyclical encoding)
5. **Campaign-level features** (aggregated CTR, impressions, clicks)
6. **User-level features** (aggregated CTR, click history)

1.4.1 Why These Features Matter

Feature Category	Purpose	Example
Exposure count	Core metric for fatigue analysis	`exposure_count=5` means 5th impression
Recency	Capture time effects	`hours_since_last=24` hours ago
Campaign-level	Control for campaign quality	`campaign_overall_ctr=0.35`
User-level	Control for user engagement	`user_overall_ctr=0.40`
Temporal	Control for time-of-day effects	`hour_of_day=14` (2pm)

[5]:
```python
# Column names for Criteo Attribution Dataset
# These match the actual column names from the dataset
user_col = 'uid'  # User identifier
campaign_col = 'campaign'  # Campaign identifier
time_col = 'timestamp'  # Timestamp (converted to datetime in previous notebook)
click_col = 'click'  # Click label (primary target)
# Alternative: use 'conversion' for conversion analysis

# Create all features
windows_hours = config['feature_engineering']['exposure_windows']
df_features = create_all_features(
    df,
    user_col=user_col,
    campaign_col=campaign_col,
    time_col=time_col,
    click_col=click_col,
    windows_hours=windows_hours
)

print(f"\nFeature engineering complete!")
print(f"Original columns: {len(df.columns)}")
print(f"New columns: {len(df_features.columns)}")
print(f"\nNew feature columns:")
new_features = [col for col in df_features.columns if col not in df.columns]
for feat in new_features[:20]:
    print(f"  - {feat}")
if len(new_features) > 20:
    print(f"  ... and {len(new_features) - 20} more features")
```

```
Computing exposure counts…
Computing time-windowed exposures…
Computing exposure recency…
Computing exposure intensity…
Computing temporal features…
Computing campaign features…
Computing user features…
Feature engineering complete. Final shape: (500000, 59)

Feature engineering complete!
Original columns: 27
New columns: 59

New feature columns:
  - exposure_count_24h
  - exposure_count_168h
  - exposure_count_720h
  - hours_since_first_exposure
  - hours_since_last_exposure
```

```
- avg_hours_between_exposures
- days_since_first
- exposures_per_day
- recent_exposure_rate
- hour_of_day
- day_of_month
- hour_sin
- hour_cos
- dow_sin
- dow_cos
- month_sin
- month_cos
- campaign_total_impressions
- campaign_total_clicks
- campaign_overall_ctr
… and 12 more features
```

1.5 Step 2: Save Feature-Engineered Data

The feature-engineered dataset is saved for use by subsequent analysis notebooks (04, 05, 06).

Output file: `../data/processed/data_with_all_features.csv`

This file contains: - Original 22 columns from the Criteo dataset - ~35 new engineered features - Total: ~57 columns, 500,000 rows

[6]:
```
# Save feature-engineered data
output_path = '../data/processed/data_with_all_features.csv'
df_features.to_csv(output_path, index=False)
print(f"Feature-engineered data saved to {output_path}")
print(f"Shape: {df_features.shape}")

# Display summary statistics for key features
if 'exposure_count' in df_features.columns:
    print(f"\nExposure count statistics:")
    print(df_features['exposure_count'].describe())

if 'hours_since_first_exposure' in df_features.columns:
    print(f"\nHours since first exposure statistics:")
    print(df_features['hours_since_first_exposure'].describe())
```

```
Feature-engineered data saved to ../data/processed/data_with_all_features.csv
Shape: (500000, 59)

Exposure count statistics:
count    500000.000000
mean          2.762458
std           5.942473
min           1.000000
```

```
25%           1.000000
50%           1.000000
75%           2.000000
max         196.000000
Name: exposure_count, dtype: float64

Hours since first exposure statistics:
count    500000.000000
mean         73.458085
std         147.507331
min           0.000000
25%           0.000000
50%           0.000000
75%          68.400139
max         739.847500
Name: hours_since_first_exposure, dtype: float64
```