Movie Rating Prediction

Senior Data Scientist Take-Home Challenge

Salvador Barcenas Valladolid

Github repository

Problem Statement

- Predict whether a user will rate a movie ≥ 4
- Binary classification problem
- Dataset: MovieLens 20M
- Main challenge: time-dependent data and data leakage prevention

Data overview

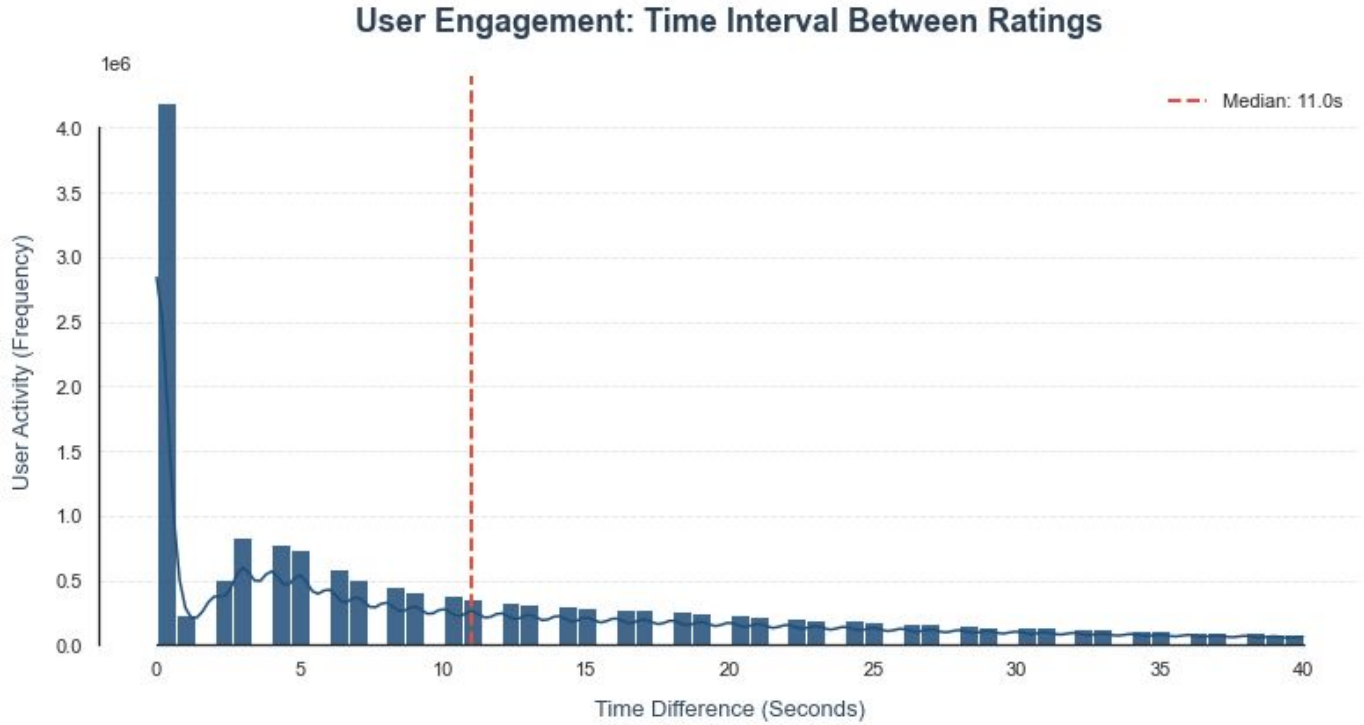
- 138,493 users
- 27,244 rated movies
- 20M ratings, ~38K unique tags
- Data spans from 1995 to 2015

Key Challenges

- Strong temporal dependency between interactions
- High risk of data leakage when building historical features
- Cold-start scenarios for users and movies

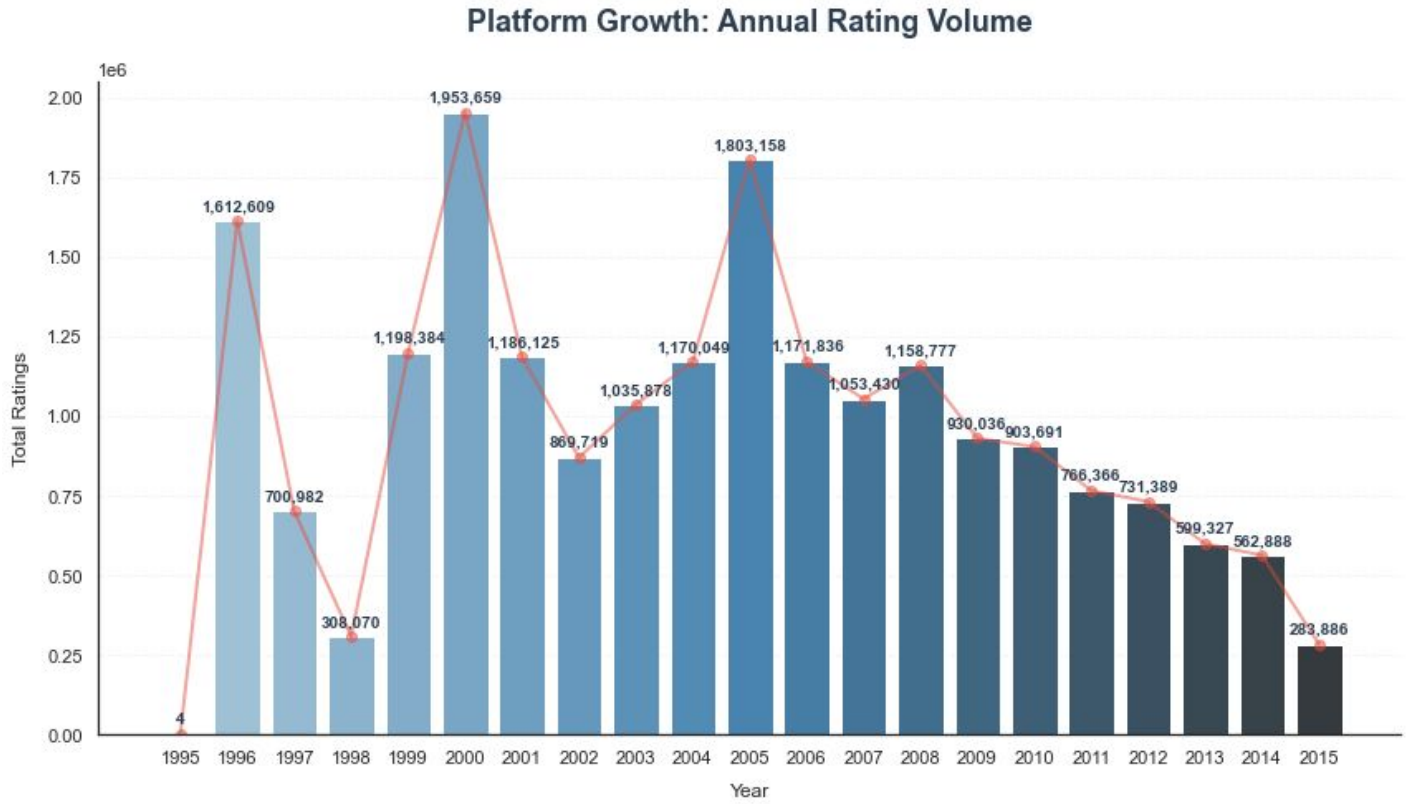
Highlights

- Users tend to rate movies in short bursts (median gap ≈ 11 seconds)
- Online predictions are triggered by user requests, not each interaction.
- All movie genres are available before the train/test split date

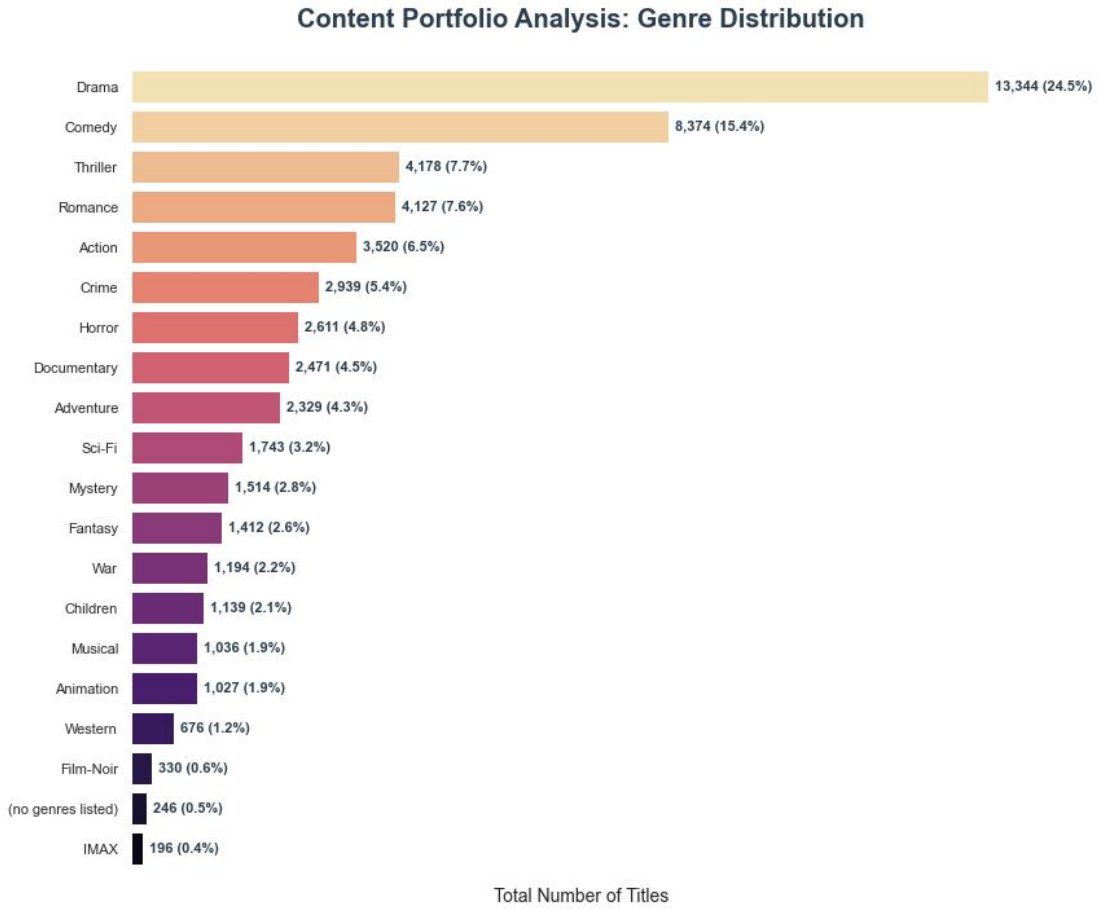


Highlights

- The number of reviews has shown a downward trend in recent years.

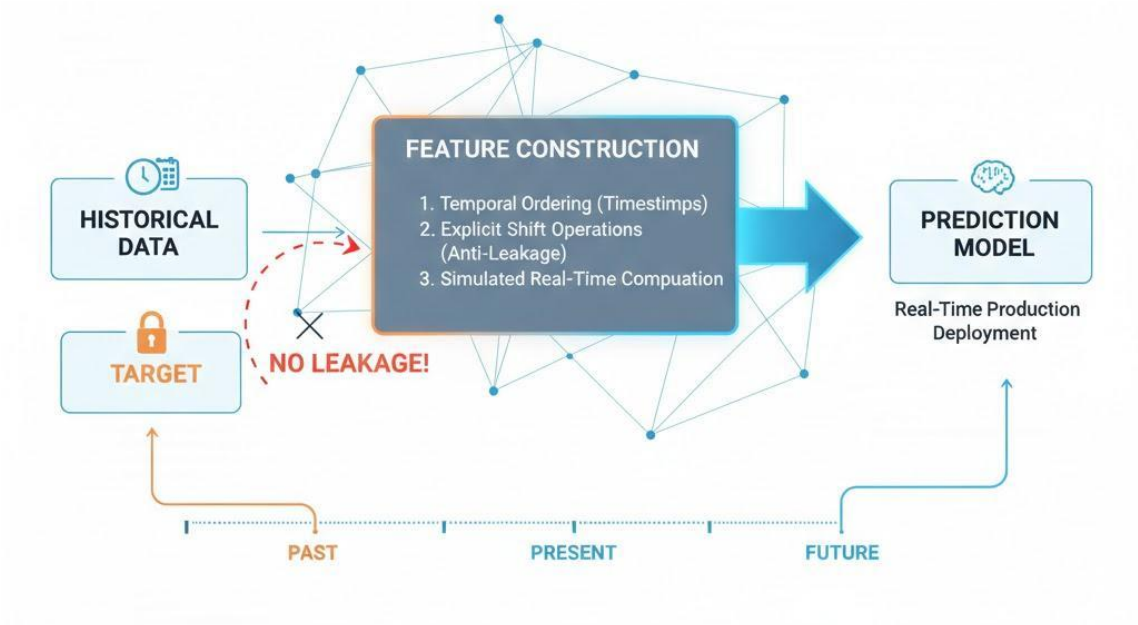


Highlights


Content Portfolio Analysis: Genre Distribution
Drama
13,344 (24.5%)
Comedy
8,374 (15.4%)
Thriller
4,178 (7.7%)
Romance
4,127 (7.6%)
Action
3,520 (6.5%)
Crime
2,939 (5.4%)
Horror
2,611 (4.8%)
Documentary
2,471 (4.5%)
Adventure
2,329 (4.3%)
Sci-Fi
1,743 (3.2%)
Mystery
1,514 (2.8%)
Fantasy
1,412 (2.6%)
War
1,194 (2.2%)
Children
1,139 (2.1%)
Musical
1,036 (1.9%)
Animation
1,027 (1.9%)
Western
676 (1.2%)
Film-Noir
330 (0.6%)
(no genres listed)
246 (0.5%)
IMAX
196 (0.4%)
Total Number of Titles

Feature Engineering Strategy

- Features are built using only historical information
- Temporal ordering enforced using timestamps
- Explicit shift operations to avoid target leakage
- Feature computation simulates real-time production constraints.



Feature engineering

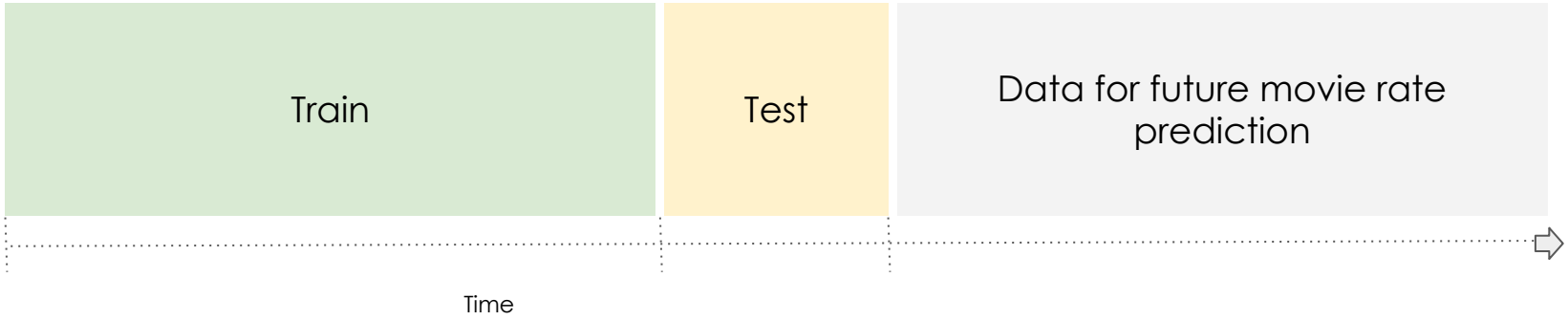
- **User behavior features:** historical average ratings, proportion of ratings ≥ 4, previous rating, number of past ratings, and time since last interaction.
- **Movie reputation features:** historical average rating, proportion of positive ratings, number of ratings, and release year, with cold-start handled via temporal ordering.
- **Tag-based features:** lowercase and deduplicated tags, sentiment scores computed using TextBlob, excluding tags after the rating timestamp.
- **Tag relevance enrichment:** tags were joined with genome_scores to incorporate tag–movie relevance weights, capturing semantic importance beyond raw tag frequency.
- **Aggregation strategy:** user, movie, and tag features merged at user–movie level using only past information; null values represent lack of history and prevent data leakage.

Preprocessing

- The following preprocessing steps were applied to the data to have it ready for training
  - Zero imputer: Replace nulls with 0's, as all variables make sense to replace them with 0
  - Max winsorizer: Cap values that are more distant from the mean by three standard deviations with this method to remove outliers

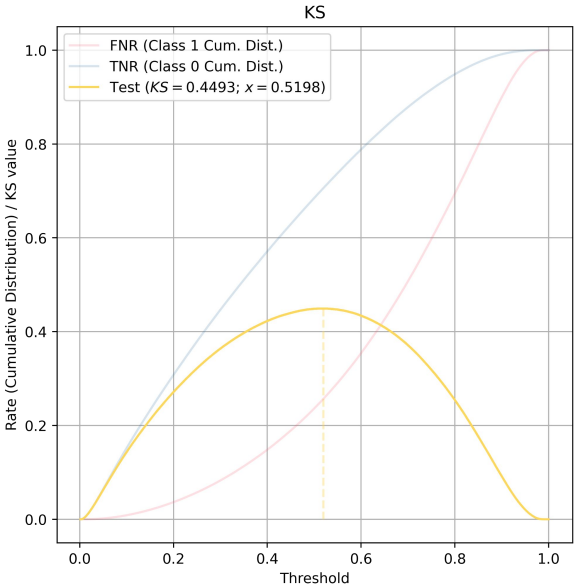
Train/test split

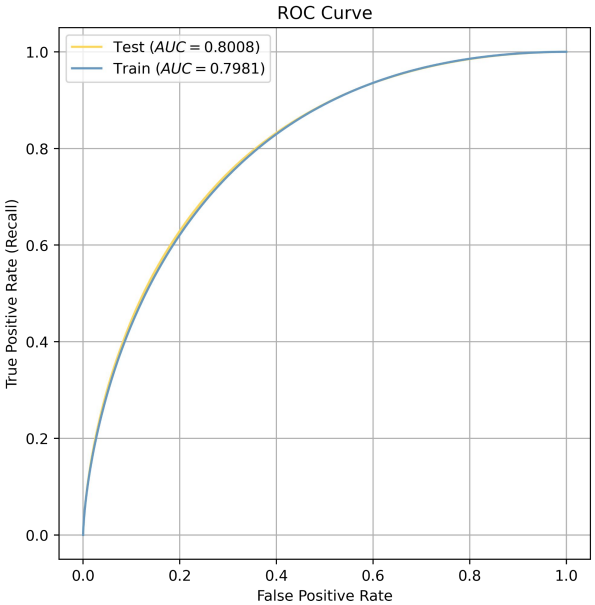
- Since we are working with a model that will be used in the future with data we haven't seen, it is preferable to split the training and test sets based on time. This way, we achieve a better evaluation compared to using a random partition where the time variable is excluded from the split
- Out-of-Time split based on timestamp
- Train data: ~85% (before 2011-01-01)
- Test data: ~15% (after 2011-01-01)

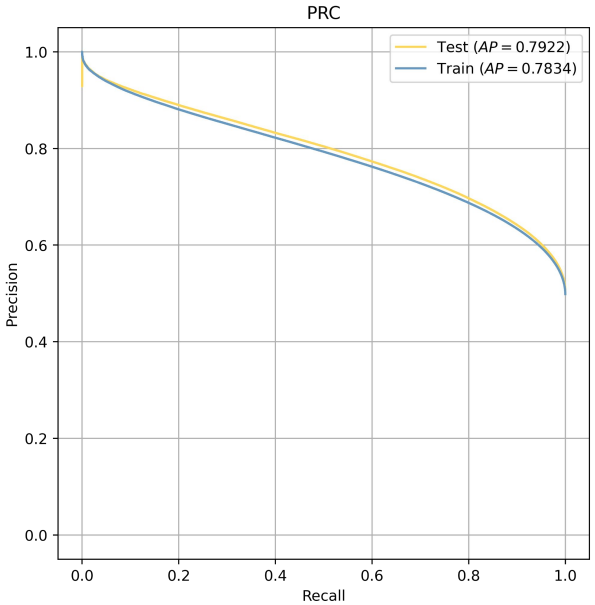


Training and evaluation

- I used an lightGBM model and Optuna OOT-style CV to find the best parameters.





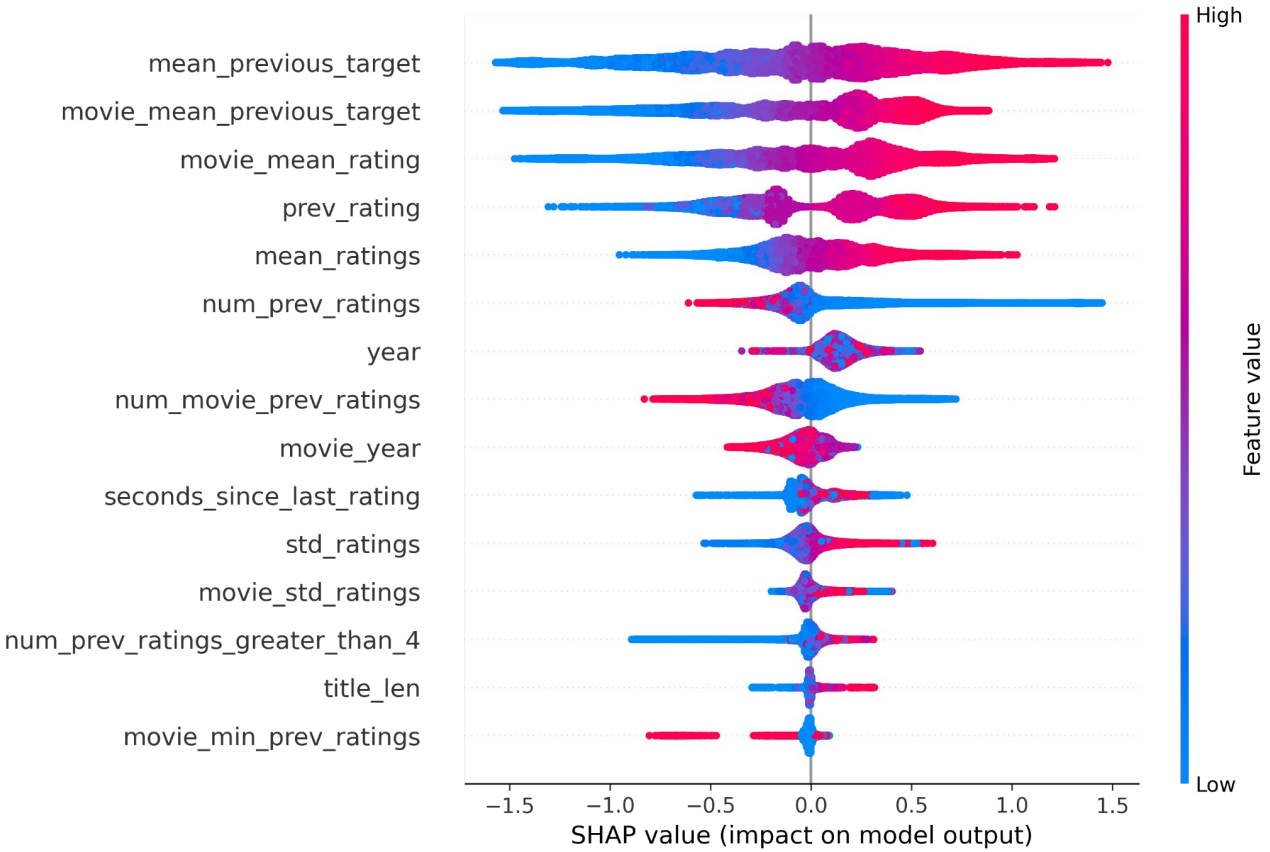


Training and evaluation

- As you can see, the performance is not spectacular; however, it is the best that could be achieved with the current features. Efforts were made to minimize overfitting.

- 'roc_auc': 0.8008
- 'threshold': 0.3704
- 'f1_score': 0.75085
- 'precision': 0.6586
- 'recall': 0.8730

Shap values

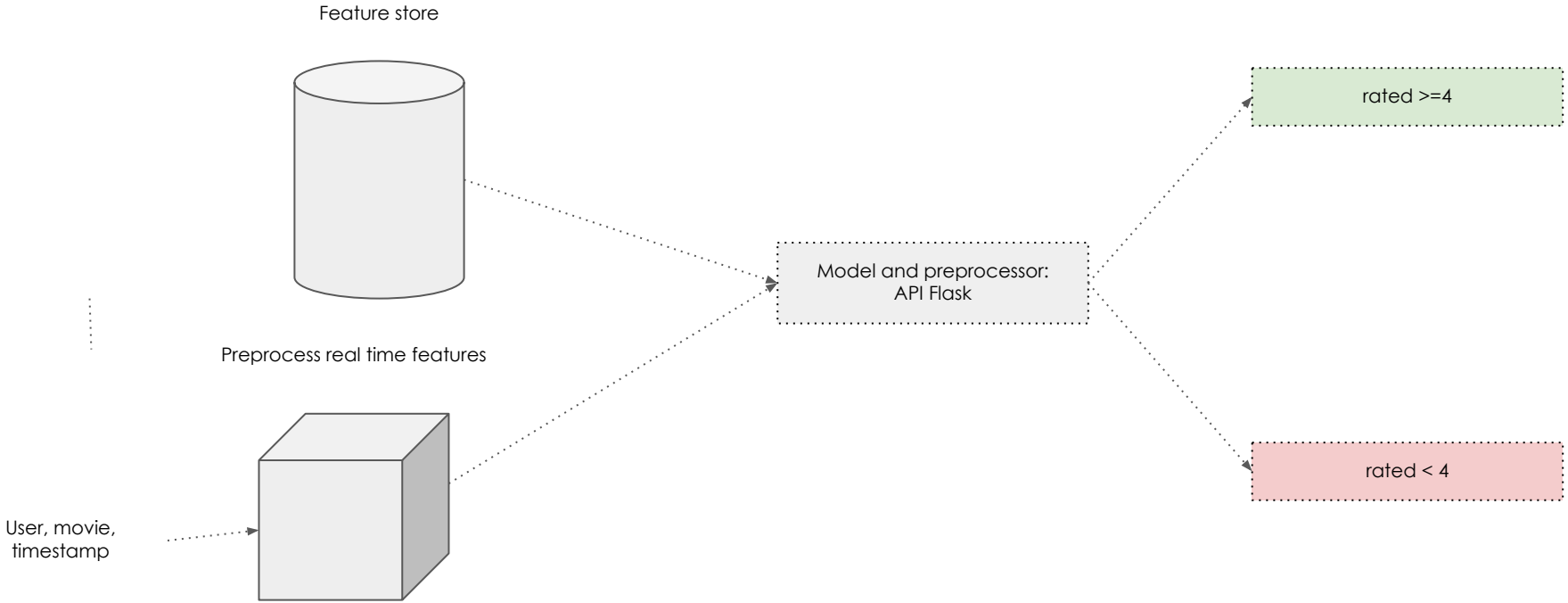

mean_previous_target
movie_mean_previous_target
movie_mean_rating
prev_rating
mean_ratings
num_prev_ratings
year
num_movie_prev_ratings
movie_year
seconds_since_last_rating
std_ratings
movie_std_ratings
num_prev_ratings_greater_than_4
title_len
movie_min_prev_ratings
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
SHAP value (impact on model output)
High
Feature value
Low

Model interpretation

- Users with a historically high proportion of ratings ≥ 4 are significantly more likely to rate the current movie positively. This behavior is consistently captured through features such as the user's average past ratings and their previous rating.
- Similarly, movies with a strong historical reputation—measured by the average proportion of ratings ≥ 4—have a higher probability of receiving positive ratings.
- Interestingly, users with a large number of past ratings tend to assign lower scores on average. This likely reflects more critical behavior due to a broader basis for comparison. A similar effect is observed for movies with many past ratings.
- Temporal effects are also present: newer movies tend to receive slightly lower ratings on average.
- Overall, the model relies primarily on the user's historical behavior and the movie's past performance, which aligns well with domain intuition and reduces the risk of overfitting to less informative features.

Model usage

- The model scores user–movie interactions in real time to predict whether the rating will be ≥ 4.
- Historical user and movie features are stored in a feature store and updated in micro-batches.
- A lightweight scoring API combines stored features with request-time context to return the prediction.



Thank You

Thank you for your time.
Questions?