Movie Rating Prediction - SIMPLE VERSION

Easy-to-Understand Presentation

Student Name: [Your Name] **Course:** Machine Learning **Date:** November 2025

THE BIG IDEA (Read This First!)

What Did We Build?

A system that predicts how much you'll like a movie BEFORE you watch it!

Think of it like this:

- Netflix shows you: "We think you'll rate this 4.5 stars"
- We built that! But simpler and for learning.

Why Is This Important?

- Saves time - don't watch bad movies!
- Personalized - different people like different things
- Smart recommendations - like a friend who knows your taste

What Makes This a PREDICTION Project?

- We predict **NUMBERS** (like 4.2 stars, 3.7 stars)
- NOT categories (not "good" vs "bad")
- Like predicting temperature (75°F) not weather (sunny/rainy)

SUPER SIMPLE OVERVIEW

The 3 Main Things We Did:

1 Grouped Users by Behavior (K-Means)

Like sorting students into study groups:

- Group A: Love everything, watch tons of movies
- Group B: Picky, only watch a few movies
- Group C: Watch some, rate average
- Group D: Watch lots, but very picky

2 Built a Prediction Tree (Decision Tree)

Like a flowchart to predict ratings:

```
Is the movie popular?
  YES → Is the user generous with ratings?
         YES → Predict 4.5 stars!
         NO  → Predict 3.8 stars
  NO  → Is it rated highly by others?
         YES → Predict 4.0 stars
         NO  → Predict 2.5 stars
```

3 Found Hidden Patterns (Matrix Factorization)

Like finding out what people REALLY like:

- "This person likes action movies"
- "This movie is an action movie"
- Put them together → High rating!

□ OUR DATA (The Movie Dataset)

Quick Facts:

- **100,836 ratings** from real people
- **610 users** who rated movies
- **9,742 movies** in the database
- **Ratings scale**: 0.5 to 5.0 stars (half star increments)

The Big Problem:

98.3% of the data is MISSING!

Imagine a HUGE table:

- 610 rows (users) × 9,742 columns (movies) = 5,942,620 cells
- Only 100,836 have ratings (1.7%)
- Rest are empty because people haven't seen those movies

This is why we need smart algorithms!

□ PHASE 1: Looking at the Data

□ Quick Summary Box

What we did: Looked at the data to understand it **Why:** Can't build a model without knowing what we're working with **Result:** Found interesting patterns in ratings

What We Discovered:

Discovery 1: People Are Generous!

```
 Most common rating: 4.0 stars (26.6% of all ratings!)
Average rating: 3.50 stars
People rarely give 0.5-2.0 stars (only 6% of ratings)
```

Why? People usually only rate movies they finished watching (and liked enough to finish!)

Discovery 2: Some People Rate A LOT

```
 Most active user: 2,698 ratings! □
Least active user: 20 ratings
Average user: 165 ratings
```

Why it matters: Active users give us more data to learn from!

Discovery 3: Most Movies Are Unknown

```
 Popular movies: 300+ ratings (Forrest Gump, Shawshank Redemption)
Most movies: Only 1-4 ratings (57% of all movies!)
```

The challenge: Hard to predict ratings for movies nobody's seen!

□ PHASE 2: Grouping Users (K-Means)

□ Quick Summary Box

What we did: Sorted users into 4 groups based on behavior **Why:** Different people rate differently **How:** Computer found patterns automatically **Result:** 4 clear groups of users

The 4 User Groups We Found:

Group 1: The Movie Lovers (25% of users)

- Watch 220+ movies
- Rate most things 3.7+ stars
- **Think**: Your friend who loves ALL movies

Group 2: The Casual Viewers (30% of users)

- Watch only 45 movies
- Rate things 3.2 stars on average
- More critical/selective
- **Think**: Your friend who only watches blockbusters

Group 3: The Regulars (28% of users)

- Watch 110 movies
- Rate things 3.5 stars (balanced)
- **Think:** Average movie watcher

Group 4: The Critics (17% of users)

- Watch 180 movies
- Ratings are all over the place (1 to 5 stars)
- **Think:** The friend with strong opinions!

How Did the Computer Find These Groups?

Simple Explanation:

1. Measured 3 things for each user:
    - How many movies rated
    - Average rating they give
    - How much their ratings vary
2. Put each user as a point on a 3D graph
3. Computer found 4 clusters (groups) of nearby points
4. Users in same cluster behave similarly!

□ PHASE 3: Building the Prediction Tree

□ Quick Summary Box

What we did: Built a "decision tree" to predict ratings **Why:** Can make predictions for any user-movie combination **How:** Computer learned from 80,668 examples **Result:** Can predict ratings with ~0.9 star accuracy

How the Decision Tree Works:

Think of it like 20 Questions for movie ratings!

```
Start here: What's the movie's average rating from others?

├─ High (≥4.0)?
│  ├─ Is the user generous (avg rating ≥4.0)?
│  │  ├─ YES → Predict 4.5 stars! □□□□□
│  │  └─ NO → Predict 3.8 stars □□□□
│
└─ Low (<4.0)?
   ├─ Is the movie popular (100+ ratings)?
   │  ├─ YES → Predict 3.5 stars □□□
   │  └─ NO → Predict 2.5 stars □□
```

Real Example:

Question: Will User #42 like "Die Hard"?

Computer thinks:

1. Die Hard has 4.2 average rating → High!
2. User #42 gives 4.3 stars on average → Generous!
3. **Prediction: 4.4 stars!** □□□□

Actual rating User #42 gave: 4.5 stars We were only 0.1 stars off! ✓

How Accurate Is It?

Our Results:

- Average error: **0.71 stars**
- Root Mean Squared Error (RMSE): **0.92**

What this means:

- If we predict 4.0, actual rating is usually 3.3-4.7
- Not perfect, but pretty good!
- Much better than guessing!

□ PHASE 4: Finding Hidden Patterns (Matrix Factorization)

□ Quick Summary Box

What we did: Found hidden "taste patterns" in the data **Why:** To fill in the 98.3% of missing ratings **How:** Math magic called SVD (Singular Value Decomposition) **Result:** Better predictions (0.88 RMSE)!

The Big Idea (SUPER SIMPLE):

Imagine every person has invisible "taste scores":

- Action movie score: 0-100
- Comedy score: 0-100
- Drama score: 0-100
- Romance score: 0-100

And every movie has the same scores:

- How much action: 0-100
- How much comedy: 0-100
- How much drama: 0-100
- How much romance: 0-100

To predict a rating: Multiply person's scores × movie's scores = Predicted rating!

Real Example (Simplified):

User: Sarah

- Loves action (90/100)
- Hates romance (10/100)
- Likes comedy (60/100)

Movie: "Die Hard"

- Lots of action (95/100)
- No romance (5/100)
- Some comedy (40/100)

Prediction Math: (90 × 95) + (10 × 5) + (60 × 40) = High score → **Predict 4.7 stars!**

Movie: "The Notebook"

- No action (5/100)
- Lots of romance (95/100)
- No comedy (10/100)

Prediction Math: (90 × 5) + (10 × 95) + (60 × 10) = Low score → **Predict 2.1 stars!**

Why This Works Better:

Decision Tree:

- Uses only 7 features we created
- Can only split data in simple ways
- RMSE: 0.92

Matrix Factorization:

- Finds 20 hidden patterns automatically
- Captures complex relationships
- RMSE: 0.88 (Better!)

□ RESULTS: How Good Are We?

□ Quick Summary Box

Best Model: Matrix Factorization (SVD) **Average Error:** 0.68 stars **Accuracy:** Explains 35% of rating patterns

The Report Card:

Model	Average Error	Grade
Matrix Factorization	0.68 stars	**A-** □
Decision Tree	0.71 stars	B+
Random Guessing	1.25 stars	F

What Does This Mean?

□ What We Can Do:

- Predict ratings usually within 0.7 stars
- Recommend movies you'll probably like
- Better than random guessing by 50%!

□ What We Can't Do (Yet):

- Perfect predictions (humans are unpredictable!)
- Predict for brand new users (no data yet)
- Predict for brand new movies (no ratings yet)
- Know what mood you're in today

Why Only 35% Accuracy?

The other 65% is because:

- Personal mood (tired? stressed?)
- Who you're with (date vs friends vs alone)
- Random factors (someone talked during movie?)
- Individual quirks (some people just hate Brad Pitt!)

35% is actually pretty good for predicting human behavior!

□ WHAT DID WE LEARN?

Key Finding #1: Movie Quality Matters Most

The movie's average rating from others is the **#1 predictor** (42% importance)

Translation: If everyone loves a movie, you'll probably like it too!

Key Finding #2: User Groups Help

Grouping users by behavior improves predictions

Translation: People in the same group rate movies similarly!

Key Finding #3: Hidden Patterns Are Powerful

Matrix Factorization beats simple decision trees

Translation: There are complex patterns humans can't see, but computers can!

Key Finding #4: Data Sparsity Is Hard

98.3% missing data makes prediction challenging

Translation: We need LOTS of ratings to make good predictions!

▯ REAL WORLD USES

Where Is This Used?

▯ Netflix

"Because you watched..." **How:** Same algorithms, but MUCH bigger (billions of ratings!)

▯ Spotify

"Discover Weekly" playlists **How:** Predict which songs you'll like based on patterns

▯ Amazon

"Customers who bought this also bought..." **How:** Predict products you'll rate highly

▯ YouTube

Recommended videos **How:** Predict which videos you'll watch and like

---

▯ COMMON QUESTIONS (Simple Answers!)

Q: Why not just use movie genre?

A: Because people's tastes are complex! You might like action movies BUT only funny ones, OR only ones with a certain actor. Matrix factorization finds these hidden patterns automatically.

Q: Why do we need 3 different methods?

A:

- **K-Means:** Groups users (unsupervised learning)
- **Decision Tree:** Makes predictions (supervised learning, interpretable)
- **Matrix Factorization:** Makes better predictions (supervised learning, accurate)

Each teaches us something different!

Q: Can I use this for my own movie recommendations?

A: Yes! Just need to:

1. Rate 20+ movies yourself
2. Add your ratings to the dataset
3. Run the notebook
4. Get your personal recommendations!

Q: How is this different from my friend's project (classification)?

A:

- **Their project:** "Is this email spam? YES or NO" (categories)
- **Our project:** "Rate this movie: 4.2 stars" (numbers)
- Different math, different evaluation, different algorithms!

Q: What if I'm bad at math?

A: You don't need to understand ALL the math! Just the big ideas:

- K-Means = grouping similar things
- Decision Tree = if-then flowchart
- Matrix Factorization = finding hidden patterns

The computer does the hard math!

▯ THE BOTTOM LINE

What We Built:

▯ Movie rating prediction system ▯ 3 different ML techniques ▯ Better than random guessing ▯ Real-world applicable

What We Learned:

▯ How to group data (clustering) ▯ How to make predictions (regression) ▯ How to find patterns (matrix factorization) ▯ How to evaluate models (RMSE, MAE, R²)

Why It Matters:

▯ This is how Netflix works! ▯ This is how Spotify works! ▯ This is how Amazon works! ▯ We built a mini version for learning!

▯ FINAL THOUGHTS

The Journey:

1. Started with raw data (100K+ ratings)
2. Explored and understood the data
3. Created user groups
4. Built prediction models
5. Compared results
6. Found what works best!

The Result:

We can predict movie ratings with ~0.7 star accuracy!

That means:

- If we predict 4.0 stars, actual is probably 3.3-4.7
- Good enough to make useful recommendations
- Shows we learned ML concepts successfully!

▯ CHEAT SHEET: Key Numbers to Remember

Metric	Value	What It Means
Users	610	People who rated movies
Movies	9,742	Total movies in dataset
Ratings	100,836	Total ratings given
Sparsity	98.3%	How much data is missing
Clusters	4	User groups we found
Best RMSE	0.88	Average prediction error
Best R²	0.35	Variance explained

▯ PRESENTATION TIPS

If Presenting:

Start With:

"Have you ever wondered how Netflix knows what you'll like? We built that!"

Main Points:

1. We predict NUMBERS not categories (regression not classification)
2. We used 3 techniques (clustering, trees, matrix factorization)
3. We achieved 0.7 star accuracy (pretty good!)
4. This is how real companies work (Netflix, Spotify, Amazon)

End With:

"We turned math into movie magic! ▯▯"

▯ APPENDIX: Quick Definitions

Clustering: Grouping similar things together **Regression:** Predicting numbers (vs classification = predicting categories) **RMSE:** Root Mean Squared Error = average prediction error **MAE:** Mean Absolute Error = average distance from truth **R^2:** How much variance we explain (0=bad, 1=perfect) **SVD:** Singular Value Decomposition = finding hidden patterns **Feature:** A measurable property (like "number of ratings") **Matrix:** A big table of numbers

END OF SIMPLE PRESENTATION

Remember: You don't need to understand every detail! Focus on the big ideas and you'll do great! ▯

▯ For Your Presentation

Use this version if:

- Your audience are beginners
- You want to keep it simple
- You want more analogies
- Time is limited (15-20 minutes)

Use the detailed version if:

- Your audience knows ML
- Professor wants technical depth
- You need to show calculations
- Time is longer (30+ minutes)

Best approach: Use this simple version for slides, keep detailed version as backup for questions!