Predicting PlanetTerp Professor Ratings

CMSC 320 – HW 4, Spring 2025
Boubacar Sall

Problem & Motivation

PlanetTerp: student-submitted ratings of UMD professors

▶️ Why predict ratings?

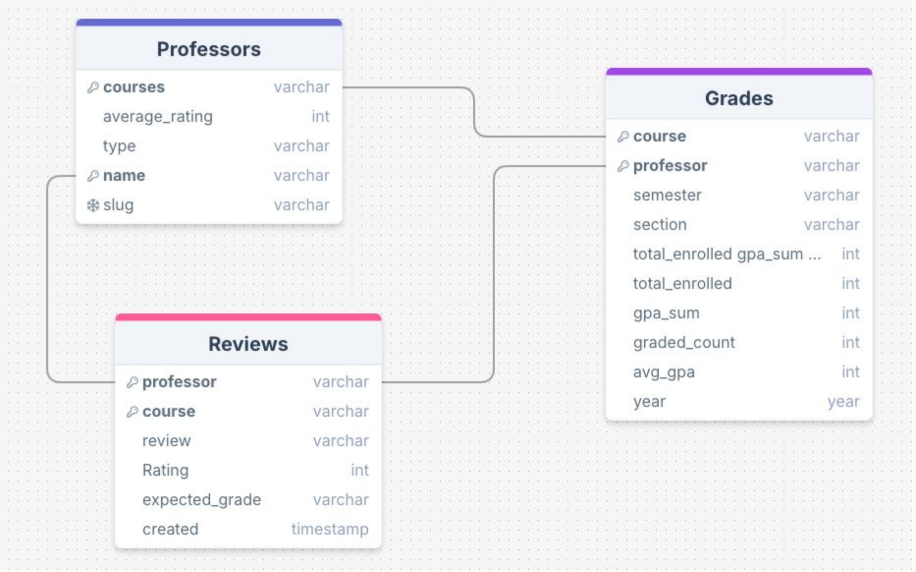
- Help students choose courses
- Potentially used to higher professors based on their qualities

▶️ Challenge: Cannot use existing average ratings directly

Data Sources & Table Creation

Extracted the data from the PlanetTerp website via API:

I split the data into 3 tables to form a database schema structure



Data Cleaning

▶️ Reviews:

- 37,401 → 37,365 reviews (drop duplicates)
- Fill missing course names ("unknown")
- Add review_word_count

▶️ Grades:

- 71,543 rows
- Aggregate total_enrolled, graded_count, avg_gpa

▶️ Professors:

- 13,421 → 4,536 professors (with 1 review)
- Compute num_courses, semesters_taught

▶️ Final Table:

- 4,538 professors

Feature Engineering

🛠️ Feature Engineering:

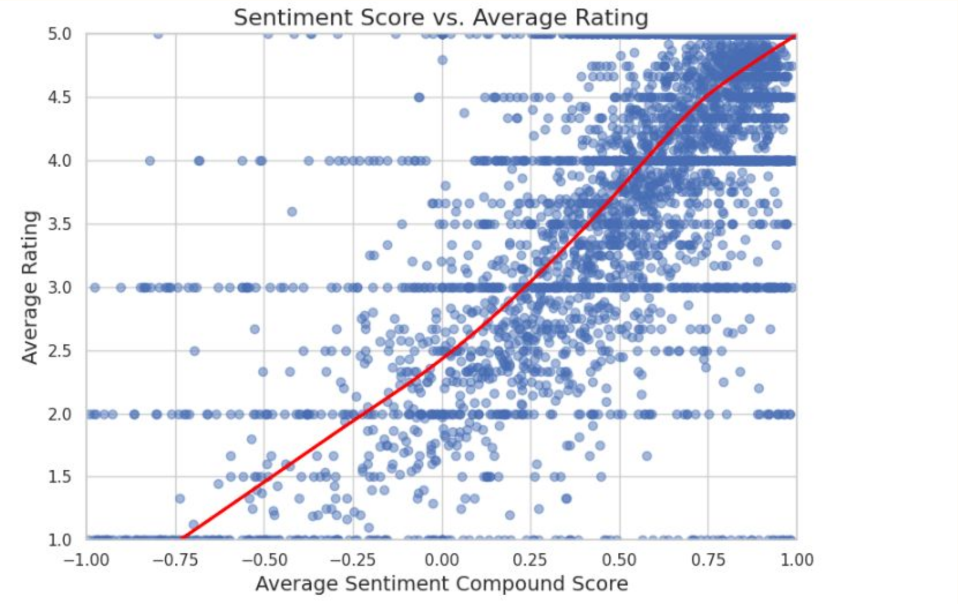
- num_courses — number of distinct courses taught
- num_semesters_taught — semesters with recorded teaching
- review_count — number of student reviews
- avg_expected_grade — mapped to numeric scale
- avg_review_word_count — average review length
- total_enrolled_prof — total students enrolled across courses
- avg_gpa_prof — average GPA from grades table

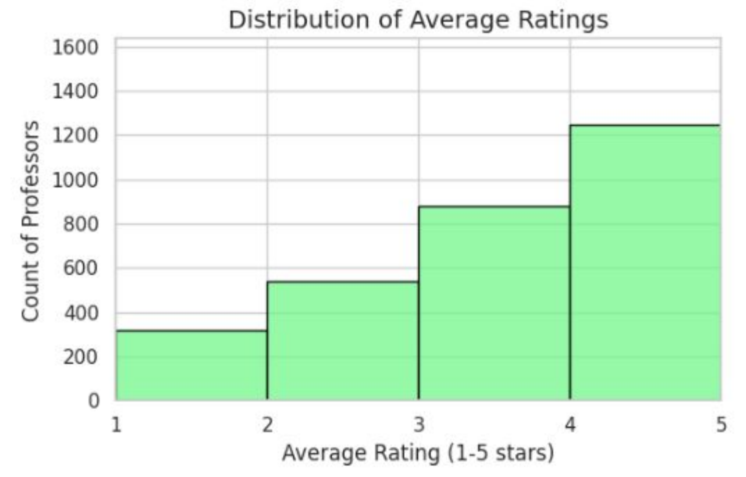
Sentiment

🧠 Sentiment Analysis:

- Used VADER to score review text (-1 to $+1$)
- Aggregated average sentiment per professor

Exploratory Data Analysis





Modeling Approaches

🧩 Models Used:

- K-Nearest Neighbors (KNN)
- Random Forest
- XGBoost

🔎 Why These Models?

- Balance between simplicity, power, and speed

Model Evaluation Setup

🧪 Data Splitting:

- 70% Training Set
- 30% Test Set (held out for final evaluation)

🔄 Validation Strategy:

- 10-Fold Cross Validation
- Training set split into 10 folds
- Train on 9 folds, validate on 1 fold (repeated 10×)
- Used for hyperparameter tuning

🎯 Final Evaluation:

- After tuning, best model evaluated on full test set
- Metrics: RMSE (error), R^2 (variance explained)

Hyperparameter Tuning

⚙️ GridSearchCV Process:

- Defined a range of possible hyperparameters for each model
- Systematically tried all combinations across the grid
- Used 10-Fold Cross Validation to evaluate each combination
- Selected the hyperparameters that minimized RMSE on validation folds

Model Results

```
KNN → RMSE: 0.697, R²: 0.635
Random Forest → RMSE: 0.669, R²: 0.664
XGBoost → RMSE: 0.653, R²: 0.680
```

🎯 Best Model:

- XGBoost achieved the lowest RMSE and highest R^2

Interpretation:

RMSE, meaning on average its predictions were off by about 0.65 stars on a 5-star rating scale.

Most important Features

	Feature	Random Forest Importance	XGBoost Importance
3	avg_sentiment_compound	0.829764	0.752410
1	avg_review_word_count	0.116762	0.116428
0	avg_expected_grade	0.046171	0.109970
2	avg_gpa_prof	0.007302	0.021192

Conclusions

- XGBoost achieved the best performance:
  - Average prediction error of ~0.65 stars
  - Explained 68% of variance in professor ratings ($R^2 = 0.680$)

- Key Features:
  - avg_sentiment_compound was the most important predictor
  - avg_expected_grade, avg_review_word_count, avg_gpa_prof also contributed

- Overall:
  - Combining sentiment analysis with structured data successfully improved rating predictions

Next Steps

🚀 • Enhance Sentiment Analysis:

– Fine-tune using BERT models on review text for deeper language understanding

• Feature Engineering:

– Add course-level variables (e.g., difficulty ratings, course size)

• Deployment:

– Build a live API/dashboard to predict ratings for new professors