w

Predicting Professor Reviews from Text

A Case Study in Balanced Classification with DistilBERT

Name: Harsh Bhati

Date: 1st December 2025

Why Does This Matter?

The Problem I Wanted to Solve:

Every semester, thousands of students write reviews about their professors on platforms like **Planet Terp**. These reviews contain rich information about teaching quality, but reading through hundreds of reviews is time-consuming.

Our Question

Can a machine learning model **understand** the sentiment in a review well enough to predict its star rating?

If yes, this could enable:

- Automatic detection of concerning feedback patterns
- Understanding what language correlates with positive vs. negative experiences
- A foundation for more advanced educational analytics

What Exactly Am I Doing?

Task: Given only the *text* of a professor review, predict the star rating (1-5) the student gave.

Example Input:

> *"Nelson is a great guy! His personality is so nice and he really cares about his students. The projects were challenging but fair."*

Expected Output: 4 stars

The model learns to associate specific words and phrases with different rating levels based on patterns in the training data.

This is Hard Because...

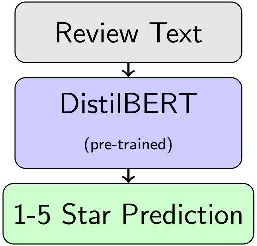
- Sarcasm exists
- Mixed reviews ("great prof, terrible exams")
- Subtle language differences between 3-star and 4-star
- Students write very differently

Our Approach: Transfer Learning with Transformers

The Big Idea: Instead of training a model from scratch, I start with a model that *already understands English* and teach it our specific task.

I used DistilBERT:

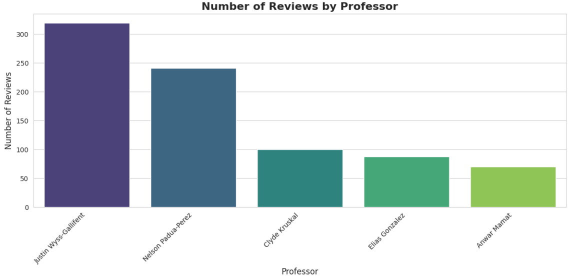
- Pre-trained on **billions of words** from Wikipedia and books
- Already knows grammar, word meanings, and context
- I just need to teach it: "this type of language = this star rating"



Why not train from scratch?

- I only have 818 reviews (way too small)
- Pre-training took Google **weeks on TPUs**
- Transfer learning lets us benefit from that work

Data: What I Collected



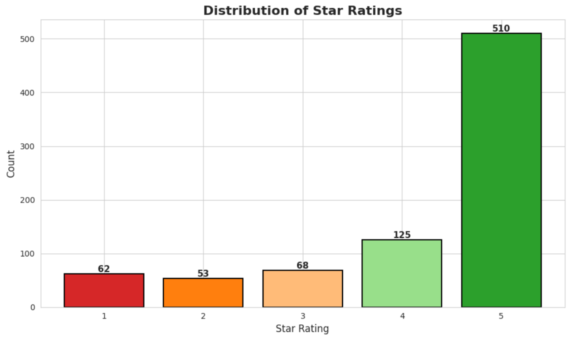
Data Source: Planet Terp API

I collected reviews from **5 popular UMD CS professors** to ensure variety in teaching styles and student experiences:

- Justin Wyss-Gallifent (319 reviews)
- Nelson Padua-Perez (241 reviews)
- Clyde Kruskal (100 reviews)
- Elias Gonzalez (88 reviews)
- Anwar Mamat (70 reviews)

Total: 818 reviews with star ratings from 1-5

The Challenge: Severely Imbalanced Data



I immediately noticed a problem:

62% of all reviews are 5-stars!

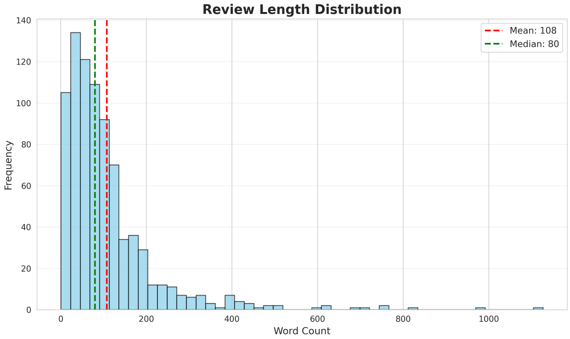
This creates a serious issue:

The Lazy Model Problem

A model can achieve **62% accuracy** by just predicting "5 stars" for everything, without learning anything useful!

5-star reviews are 9.6x more common than 2-star reviews. The model will be heavily biased toward the majority class.

Understanding the Reviews



What do the reviews look like?

- **Average length:** 108 words
- **Most reviews:** 40-130 words
- **Some outliers:** Up to 1,127 words!

Design Decision:

I set the maximum input length to **256 tokens**, which captures approximately 90% of reviews in full.

Longer reviews get truncated, but the most important sentiment usually appears early ("This professor is amazing..." or "Worst class ever...").

Our Experiments: A Systematic Approach

I ran **three experiments** to systematically improve performance:

Experiment 1: Baseline

3 epochs, no modifications

Our starting point. Train the model with default settings and see what happens.

Goal: Establish a performance floor

Experiment 2: More Training

5 epochs, no modifications

Maybe the model just needs more time to learn? Let's give it more passes through the data.

Goal: Test if longer training helps

Experiment 3: Fix Imbalance

5 epochs + class weights

Penalize the model more for misclassifying rare classes (1-4 stars).

Goal: Address the imbalance

How Does Class Weighting Work?

The Intuition: Make the model "care more" about getting rare classes right.

Without weighting:

- Misclassify a 5-star review: small penalty
- Misclassify a 2-star review: small penalty
- Model learns: "just predict 5-stars, it's usually right"

With weighting:

- Misclassify a 5-star review: small penalty
- Misclassify a 2-star review: **LARGE penalty**
- Model learns: "I need to identify ALL classes correctly"

Weight Formula

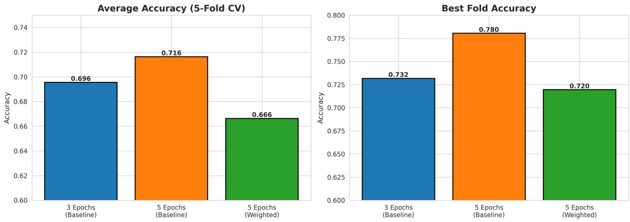
$$w_c = \frac{N}{C \times n_c}$$

Where:

- N = total samples (818)
- C = number of classes (5)
- n_c = samples in class c

2-star (53 samples) gets **9.6x higher weight** than 5-star (510 samples)

Results: The Big Picture



Test Set Accuracy:

Experiment	Accuracy
Baseline (3 ep)	74.8%
Extended (5 ep)	81.6%
Weighted (5 ep)	**92.6%**

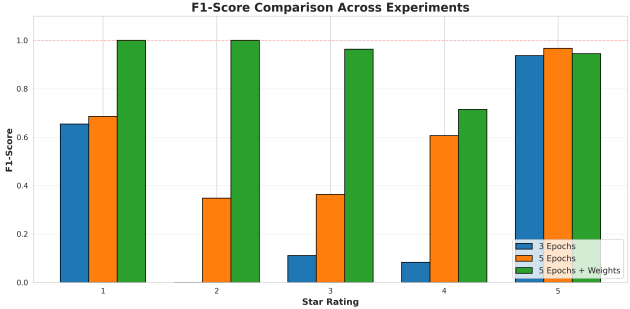
Left chart: Average accuracy across all 5 folds during cross-validation. The weighted model shows lower CV accuracy because it optimizes for balanced performance.

Right chart: Best single fold accuracy, showing each approach's peak potential.

Key Finding

Class weighting improved test accuracy by **+17.8 percentage points** over baseline!

The Real Story: Per-Class Performance



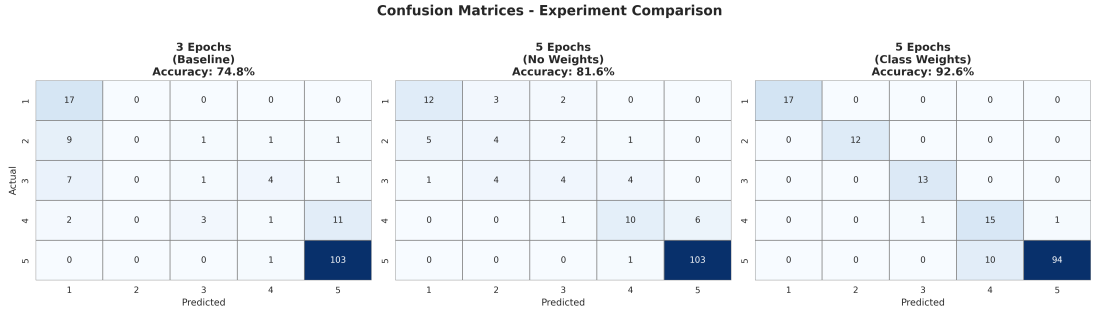
This is where it gets interesting.

Overall accuracy hides the real problem. Look at F1-scores per class:

Class	Before	After
1-star	0.65	**1.00**
2-star	0.35	**1.00**
3-star	0.11	**0.96**
4-star	0.08	**0.71**
5-star	0.93	0.95

The baseline model **completely failed** on 3-star and 4-star reviews!

Visualizing the Improvement: Confusion Matrices



What I see:

- **Left (Baseline):** The model confuses neighboring classes frequently. Reviews get misclassified to adjacent star ratings (e.g., 4-stars predicted as 5-stars).
- **Right (Weighted):** Strong diagonal pattern shows predictions matching true labels. The model learned to correctly distinguish between all rating levels.

Why Did Class Weighting Work So Well?

Let's understand what happened:

Without Weighting

The model saw 510 five-star reviews but only 53 two-star reviews during training.

It learned: "When in doubt, predict 5 stars, that's right 62% of the time!"

Result: High accuracy on 5-stars, near-zero on everything else.

With Weighting

Each 2-star review now counts 9.6x more in the loss function.

The model *must* learn what makes a 2-star review different, or it gets heavily penalized.

Result: Balanced performance across ALL classes.

The takeaway: In imbalanced datasets, raw accuracy is misleading. You must look at per-class metrics and address the imbalance directly.

Conclusions: What I Learned

Finding 1: Transfer Learning Works

DistilBERT effectively learned sentiment from just 818 reviews, achieving **92.6% test accuracy**.

Finding 2: Class Imbalance Matters

Without weighting, the model achieved 74.8% by mostly predicting 5-stars **(failing on minority classes)**.

Finding 3: Weighting Transforms Results

Minority-class F1 improved dramatically:
4-star: **0.08 → 0.71**, 3-star: **0.11 → 0.96**

Finding 4: More Epochs Help (A Bit)

Increasing epochs 3→5 gave +7%, but **class weighting contributed twice as much** improvement.

Final Result: 92.6% accuracy with balanced predictions across all star ratings

Future Work & Limitations

Limitations of Our Study:

- Only 818 reviews (relatively small dataset)
- Only 5 professors from CS department

Future Directions:

- Collect more data across departments
- Try larger models (RoBERTa, BERT-large)
- Data augmentation for minority classes
- Analyze *which words* predict each rating
- Deploy as a real-time analysis tool

Practical Applications

This model could help students quickly gauge professor sentiment, enable departments to identify feedback trends, and provide researchers with tools to study educational quality at scale.

Thank You!