Final Project

Hugh Daly, Aneesh Soni, Serena Wang

Intro

When it came to the final term project, our overall goal was to increase accuracy by implementing class concepts such as deep learning, fine tuning models for sentiment analysis, prompt engineering with GPT 4o, and even using traditional machine learning approaches. Along the way, we tried six different model approaches, experimenting with feature combinations and hyper parameter configurations resulting in about 45 total models built (see table 1 below). We ultimately found that gradient boosted techniques worked best for us.

Table 1: Model Approaches and Target Variable Predicted (best model highlighted in green)

Model Approach	Target Variable to Predict
Fine Tuned LLM	Rating Class (low/mid/high)
Zero Shot Approach with GPT-4o	Rating Class (low/mid/high)
Few Shot Approach with GPT-4o	Rating Class (low/mid/high)
Feedforward Neural Network	Rating Class (low/mid/high)
Feedforward Neural Network	Overall Rating (1-10)
Extreme Gradient Boosted (XGBoost)	Rating Class (low/mid/high)

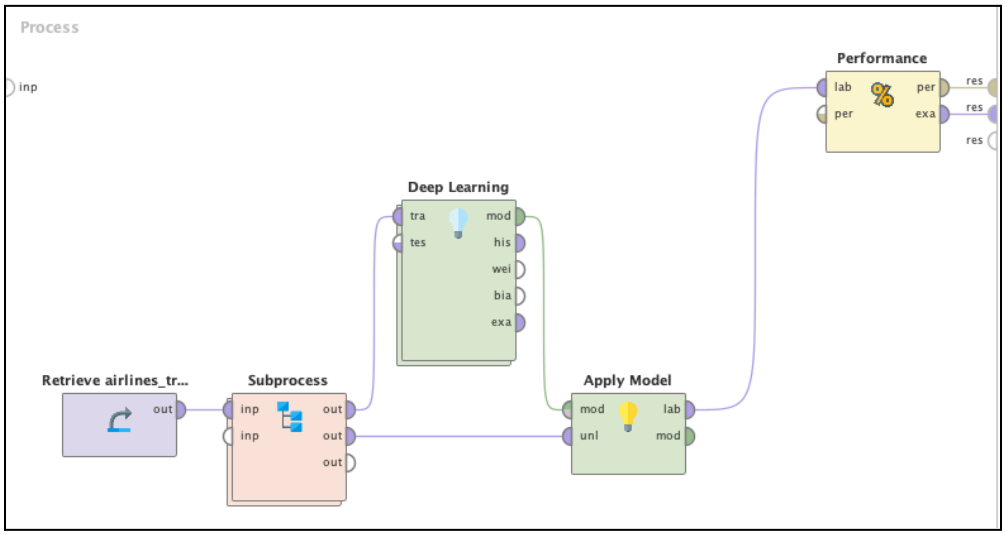
We'll first explore the initial models we developed before diving into our best performing model.

Initial Attempts: Other Models Used

Question: Describe what you have tried before your final/best submission and what you have learned from the prior experiences and past failures

Initially, we experimented with feedforward neural network models. We first created a vector embedding of length for the “Reviews” and ran a simple deep learning feedforward model on that (see image 1).

Image 1: AI Studio Process for Feedforward Neural Network using only Review Embeddings

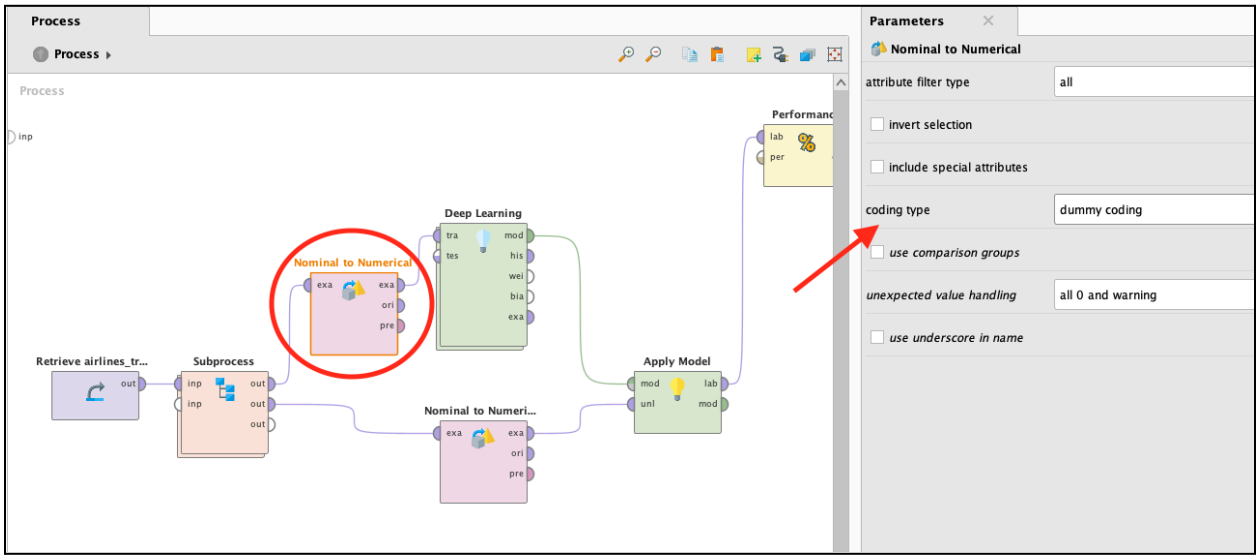


Using fairly standard hyper parameters we got promising results. In fact, our first submission on the **test dataset resulted in 79.4% accuracy.** After observing the results on the training dataset we realized that the model struggled most with classifying reviews rated as “mid”. In order to hopefully improve upon the accuracy of the model, we decided to add all 10 attributes to the model instead of solely relying on the “Reviews” vector embedding.

In order to do this, however, this meant that we needed to convert the categorical features into numerical values for easy use in our models. We did this by leveraging the “nominal to numerical” process in AI studio to one-hot encode these features (see image 2).

(CONTINUED TO NEXT PAGE)

Image 2: AI Studio Process for Feedforward Neural Network with One-hot encoding



After using all 10 attributes and the “Reviews” vector embedding we **achieved accuracy of 80.6% on the test dataset**, a solid improvement compared to our first approach which only relied on the “Reviews” vector embedding. However, we also noticed that we had a fairly large discrepancy between the accuracy of our training and testing dataset (training accuracy of 85%). As a result we iterated a few more times with this approach we eliminated certain attributes from the mix and ultimately landed on “Airline”, “Class”, “Type of Traveller” and “Verified” as having the most predictive power amongst the set of attributes.

Post feature reduction, we realized that we could also create vector embeddings for the “Title” feature as we felt that it served as a condensed version of the “Reviews”. Rerunning the model with the embedding and the finalized set of attributes we improved upon our **accuracy to 81% on the test dataset**.

At this point the only thing left for us to try for our feedforward neural network was hyper parameter tuning. We adjusted the learning rate, number of epochs, optimization method, and updater. Ultimately after several iterations of tuning these parameters we landed on the following values for each parameter which resulted in the best overall performance for our feedforward neural network:

learning rate = 0.0005
epochs = 200
optimization method = “Stochastic Gradient Descent”
updater = “Adam”

Once we used these parameters we were able to achieve an **accuracy of 81.2% on the test dataset** as submitted on gradescope. What was interesting, however, was that when we used this on the training dataset we achieved an accuracy of 87%+. Therefore, this was indicating to us that our deep learning feed forward neural network was overfitting on the training dataset and

underperforming when put against new unseen data. This is somewhat expected given that our training dataset was fairly small, only 2000 samples, and there was a potential for our model to "memorize" and ultimately overfit on certain inputs.

Due to the challenges of overfitting we briefly pivoted to a regression model that aimed to predict the overall rating and then convert that to a class of low/mid/high. However, this didn't yield positive results as we were struggling quite a bit on the training dataset, unable to get better than 65% accuracy.

The next approach we attempted was to pass the review data to OpenAI and have their GPT models do the heavy lifting for us. For this method we first tried a zero-shot approach using GPT-4o where all we did was pass the raw review data to OpenAI and had it return "low, mid, or high" for each review (see table 2 in the appendix for the prompts used). After observing the outputs, we only got a little better than 60% accuracy on the training dataset and noticed the model struggled very much with predicting "mid". Furthermore, for reviews that were lengthier and had multiple messages it would not fully understand what portion of that review it should focus on and would get these results mixed up. After this, we tried a few-shot approach where we passed in a prompt along with examples of a low, mid and high review that we selected from the dataset and felt provided a holistic example of each rating class. Surprisingly this didn't prove to be a much better approach. In fact, when we first tried this OpenAI's GPT-4o models didn't give us outputs for every review even though we explicitly asked for it. Ultimately, after a few adjustments we were able to get **72% accuracy on the training dataset.** We recognized the limitations of zero-shot and few-shot methods for nuanced tasks like rating classification. These approaches struggled to understand contextual nuances and handle ambiguous or lengthy reviews, emphasizing the importance of carefully tailoring input data and prompts for optimal outcomes.

Finally, we tried finetuning an open source model from HuggingFace. We tried predicting the rating class by only using the embeddings of the reviews. We first tried using the bert-base-multilingual-uncased-sentiment model given it was pre-trained however it didn't yield particularly promising results. As a result, this led us to trying out a few different open source models available on hugging face including one's specifically trained on shorter form text (such as twitter posts) along with other general sentiment analysis models, but the best ones resulted in about 70% accuracy on the training dataset. From this we learned that fine-tuning pre-trained models like BERT was extremely time-intensive and compute-heavy, and we found that it often led to diminishing returns when dealing with noisy or complex datasets. This experience taught us to assess whether the potential performance gains justify the additional effort and resources (ex. time consuming and heavy nature of the fine-tuning approach).

Overall, these experiences underscored the need to align our methodological choices with the specific demands and constraints of the task.

Best Performing Model

Model Development

Question: What you have done and why you designed your model this particular way?

After attempting nearly 35+ different models as discussed in the previous section we thought it might be worth traditional machine learning approaches given the size of the dataset was rather small. Additionally, since we had already done a lot of feature engineering we felt that the heavy lifting aspect of machine learning was likely behind us.

Given our previous experience working in machine learning roles, we thought it would be worth implementing an extreme gradient boosted model (downloaded from AI studios extensions marketplace “XGBoost Extension”). Gradient boosting is quite effective because it leverages tree-based decision structures to capture subtle patterns in the data. This made it particularly adept at minimizing errors in ambiguous categories, such as the "mid" rating class, where other models struggled. The design choice reflected the algorithm’s adaptability to diverse feature sets, both categorical and numerical, while maintaining robustness against noisy data. These characteristics made gradient boosting a logical and optimal choice for this project.

Using this weak learner approach the only thing we really had to adjust was the hyper parameter values for XGB similar to how we did it for our feed forward neural network previously. We experimented with the following parameters: booster, rounds, early stopping, learning rate and max depth. After a grid search approach we landed on these values for each parameter:

booster = “tree booster”
rounds = 50
early stopping = none
learning rate = 0.05
max tree depth = 6

Originally the model was underfitting on the training dataset and as a result we adjusted the learning rate inversely with rounds for slow but robust training to capture more of the characteristics of the training dataset.

Model Performance

Question: What are the performance results and how would we interpret the results?

After we tuned the parameters, we were able to achieve the following accuracy results on the training dataset (image 3 below)

Image 3: Accuracy of Extreme Gradient Boosted Model

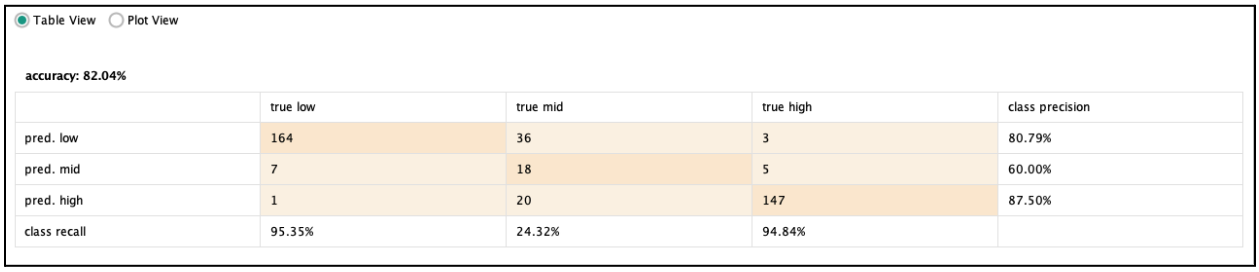


Table View Plot View

accuracy: 82.04%

	true low	true mid	true high	class precision
pred. low	164	36	3	80.79%
pred. mid	7	18	5	60.00%
pred. high	1	20	147	87.50%
class recall	95.35%	24.32%	94.84%	

This table showed an **accuracy of 82.04% on the training dataset** with incredibly high recall for the two extremes of the spectrum (i.e. low and high). While the model still struggled with accurately predicting the “true mid” data samples it did perform in a manner that seemed a bit more evenly distributed in comparison to the deep learning neural networks.

As a result, even though the training accuracy in this instance was lower than the training accuracy achieved by the deep learning approach we still put in a submission for our gradient boosted approach. Surprisingly, we achieved our best overall result with an **accuracy of 82.8% on the test dataset**! This indicated that our model was no longer excessively overfitting like with our feed forward neural network. Therefore, we were a bit more confident in any notable improvements in accuracy on the training dataset as we felt that it would translate fairly well to the test dataset.

We realized that this boosting approach excelled due to its ability to handle complex interactions and nonlinear relationships within our particular passenger ratings dataset. Since all features were already evaluated and encoded numerically it was able to effectively handle the challenges posed by noisy and imbalanced data, achieving more reliable classification across the low, mid, and high categories.

The model's performance revealed critical insights into data handling and feature importance. By focusing on the classification task, gradient boosting leveraged its capability to model nonlinear relationships and feature interactions effectively. Occasional overfitting during initial iterations was mitigated by fine-tuning parameters like learning rate and tree depth, which further enhanced generalization. The success of this approach underscored the importance of aligning model design with the specific requirements of the dataset such as data size and task structure, making gradient boosting the standout performer in our project.

Conclusion

Our key takeaways from the term project include:

- **Other approaches for small datasets:** There are several other approaches that handle smaller datasets potentially better than DL techniques
- **Noisy and Categorical Features:** The original dataset contained lots of features that we felt weren't necessary for training purposes. We ultimately reduced the dimensionality of the data

- **Overfitting on Training Data was an Issue:** We spent a lot of time tuning and testing our model on the training dataset to improve accuracy but the results on the test dataset drastically underperformed vs the training data

Appendix

Table 2: Prompts used for GPT-4o model

Approach	Prompt
Few-Shot Prompt used for GPT-4o model	You are an expert in sentiment analysis. Analyze the following airline passenger review and categorize it as one of three categories: high, mid, or low. Base your categorization on the overall tone, context, and details of the review. high: The review conveys an excellent customer experience with strong positive sentiment. The passenger is highly satisfied, with minimal or no significant complaints. mid: The review reflects a mixed experience, with both positive and negative elements. The passenger is moderately satisfied but mentions notable drawbacks. low: The review indicates a poor customer experience, with significant dissatisfaction, major complaints, or frustration that outweighs any positive elements. Examples: Input Review: "Night flight Amsterdam - Dubai followed by a short hop to Muscat. Slept very well on the A380, good connection in DXB and then a 46 min flight to Oman. I like Emirates' consistent level of service. Their pricing remains sharp, they really connect the world. Not to forget their excellent website: so easy to manage one's booking, reserve seats, upgrade using miles. Food the only item to improve." Output: high Input Review: "Hong Kong to Paris via Dubai with Emirates. Had great 6 days stopover in Dubai. Value for money! The best entertainment system on board for sure. Loved their live TV, but unfortunately it is not available on all flights. Meals are just okay, service is slow and disorganized. They take out blankets and headsets from the people 40 minutes before landing, while people moving to the toilets up and down aisles and it creates such a mess in the cabin." Output: mid Input Review: "To check in I found the economy class queue was clearing much quicker, they said the business lounge was closed and gave me $70 voucher that no shop in the Sydney airport food court accepted, then they send me on a wild goose chase to find their business lounge in Qatar as no one knew where it was located and was sent on multiple occasions to the wrong lounge. The quality of my food was disgusting with limited options for vegetarians." Output: low Your output must strictly be one of the three categories and there must be an

	output for every input: high, mid, or low Review: [[Reviews]] Rating Class:
Zero-Shot Prompt used for GPT-4o model	You are an expert in sentiment analysis. Analyze the following airline passenger review and categorize it as one of three categories: high, mid, or low. Base your categorization on the overall tone, context, and details of the review. high: The review conveys an excellent customer experience with strong positive sentiment. The passenger is highly satisfied, with minimal or no significant complaints. mid: The review reflects a mixed experience, with both positive and negative elements. The passenger is moderately satisfied but mentions notable drawbacks. low: The review indicates a poor customer experience, with significant dissatisfaction, major complaints, or frustration that outweighs any positive elements. Review: [[Reviews]] Rating Class: