CSC311: Introduction to Machine Learning

Project Proposal

Submitted by

Wilton Miller (1009976171), Benjamin Gavriely (1009970015), Christopher Marrella (1008277152)

University of Toronto
Fall 2025

1 Data Exploration

1.1 Dataset Summary

The dataset contains 11 columns, and a total of 825 student responses. The result is a categorical value form $\{ChatGPT, Claude, Gemini\}$, each target equally has a $\sim$ 33% distribution among the data set. The 9 features that make up the responses consist of, 3 open-ended text responses, 4 numerical responses (1-5 scale) and 2 multi-select categorical fields (comma separated).

1.2 Distributions

For the numerical features, i.e. the responses on a 1-5 scale, the distributions across all three models are a follows:

1. How likely are you to use this model for academic tasks?

3 — Neutral / Unsure	224
4 — Likely	223
2 — Unlikely	140
1 — Not at all likely	139
5 — Very likely	99

2. Based on your experience, how often has this model given you a response that felt suboptimal?

3 — Sometimes	411
2 — Rarely	177
4 — Often	152
1 — Never	60
5 — Very often	24
NaN	1

3. How often do you expect this model to provide responses with references or supporting evidence?

3 — Sometimes	261
2 — Rarely	199
4 — Often	154
1 — Never	99
NaN (High # caused by data inconsitiency)	62
5 — Very often	50

4. How often do you verify this model's responses?

4 — Often	222
3 — Sometimes	218
2 — Rarely	157
5 — Very often	131
1 — Never	93
NaN	4

For the multi-select categorical features, the top 3 choices per feature are as follows:

1. Which types of tasks do you feel this model handles best? (Select all that apply.)

• Explaining complex concepts simply	462
• Writing or debugging code	390
• Drafting professional text	375

2. For which types of tasks do you feel this model tends to give suboptimal responses? (Select all that apply.)

• Math computations	465
• Writing or debugging code	358
• Data processing or analysis	287

1.3 Data Issues and Preprocessing

Some of the open ended response cells are left blank, or contain placeholder strings such as #NAME? (there are 4 instances of such). The majority of the blank cells in the responses appear to arise because a specific student has given the same answer across all three models, the first cell will contain the answer itself with the other two left blank to signify that it is the same. This pattern is also used in the Supporting Evidence column, resulting in a high number of NaN responses. These structured blanks will be filled within each *student_id* group by propagating the non-empty response using forward and backward fill. The remaining empty text fields will be replaced with an empty string so they can be compatible with any text vectorization methods. For the numerical responses other than the Supporting Evidence instance, missing values will be filled with the neutral midpoint (3), this keeps consistency in the data while minimizing distortion of the overall distribution.

After the data has been cleaned, the features will be transformed as follows. The text responses will be cleaned (converted to lowercase, punctuation removed) and represented using TF-IDF vectors, which weight distinctive words more heavily than common ones. The 1-5 scale responses will be converted to a numeric form and normalized to [0,1]. The multi-select categorical columns, that have the comma separated tasks, will be split in commas and one-hot encoded to create binary indicators for each distinct task. These preprocessing steps produce a clean dataset suitable for various models.

1.4 Data Splitting and Insights

Data will be split using the *student_id* column. Grouped splits will be done so all 3 answers for on specific student will stay in the same set, i.e. no leakage across training, validation and test. We will assign 80% of the data to training and 20% to test. Our internal test set will only be used for the final evaluation of model performance. NOTE: the data exploration being performed has been done on the entire CSV file, and is independent of the split. So no exploration will be performed on the test set once we have split.

Inspection of the open-ended text shows clear vocabulary differences among labels: words such as “math,” “accurate,” and “efficient” occur frequently in **ChatGPT** responses, while **Claude** responses often include “creative,” “thoughtful,” and “verbose.” Similar patterns also exist in the 1-5 scale responses. After reviewing the data using pandas, trends such as higher “likelihood to use for academic tasks” replies correlate more with **ChatGPT**.

2 Methodology

2.1 Model Families

Our choices of models are based on the fact that this is a multi-classification problem. Additionally, we work between different tradeoffs in the models because of the mix of text and numeric features.

1. **Multiclass Logistic Regression**: A linear baseline model that predicts a class using softmax, we can use feature weights for words and numerical features. This may not be ideal if the data is not linearly separable, but can act as a strong baseline especially with the way that we are using TF IDF.
2. **Decision Tree Classifier**: This model can still make predictions on numerical features despite not being as powerful for text features. We still would like to use it because it helps make the dataset more interpretable.
3. **Feed-forward Neural Network**: This can process all our types of features and can deal with linearly inseparable data as well. However, this model can be less interpretable and unnecessary depending on the accuracy of our other choices.

2.2 Optimization techniques

For our decision tree model, we will use gini impurity optimization to choose splits. For our other two models, we will use Adam as an optimizer. Because Adam adapts the learning rate, and our model will not be particularly large, we will not be using a learning rate scheduler. We will also use an L2 regularizer in our cost function.

2.3 Validation method

Our data will be split into 80% training and 20% test. This test set will be only used once at the end to evaluate model performance. To validate our model, we chose to use k-fold cross-validation on the training set because of the limited number of data points that we have. For the same reason, we will split our training set into 10 folds, which should give us a better estimate of performance, and not require too much computation due to the data set size.

2.4 Hyperparameters

1. Hyperparameters for Logistic Regression

- **Regularization strength C:** Logistic Regression is a linear model, and regularization controls how much it penalizes large coefficients. Adjusting C helps you find the right balance between underfitting and overfitting.

2. Hyperparameters for Decision Tree
   - **Max Depth:** A shallow tree may miss patterns, while a deep tree can overfit. Tuning max depth helps balance bias and variance.
   - **Min samples split:** Larger values make the tree more conservative, reducing overfitting. Smaller values allow more splits, capturing more detail in the data.
   - **Min samples leaf:** Prevents leaves from having very few samples, which improves generalization and model stability.
   - **Max features:** Encourages diversity in splits, reduces overfitting, and improves efficiency, which is important when dealing with text features.
3. Hyperparameters for Neural Network
   - **Hidden Layers:** It will allow the model to learn complex, non-linear relationships.
   - **L2 weight decay:** Prevents overfitting by keeping weights small and improving generalization.
   - **Batch size:** Balances speed and stability.

2.5 Model Evaluation

To evaluate model performance, we used accuracy and F1 score as the main metrics, along with additional checks for precision, recall, and confusion matrices. Accuracy provided a quick measure of overall correctness. F1 score was chosen because it balances precision and recall equally across all the classes, giving each class the same importance regardless of frequency. This makes it fairer to measure performance when the dataset is imbalanced.