CSC311: Introduction to Machine Learning

Project Report

Submitted by

Roy Gal, William Gao, Zehao Peng, Steven Qiao

University of Toronto
Fall 2025

1 Executive Summary

We explored random forest classifiers, naive Bayes classifiers, and multinomial logistic regression models. Our best-performing model was a random forest classifier, with a projected accuracy of 77%. The random forest's ability to model complex, likely nonlinear relationships between a mix of categorial and text features makes it suitable for this task.

2 Data Exploration

2.1 Summary of dataset

Each of 275 students contributed one data point for each class label (ChatGPT, Claude, Gemini), resulting in a perfectly balanced 3-class dataset. The features include:

4 numerical features: likelihood of academic use, frequency of suboptimal responses, frequency of references, frequency of user verification.

2 categorical features: most suitable tasks, least suitable tasks.

3 text features: most suitable tasks, least suitable task with explanation, method of user verification.

The distribution of each of numerical feature is shown in Figure 1. As expected, the mean and median are generally close to the neutral answer 3, although the mode is 4 for the first and last questions.

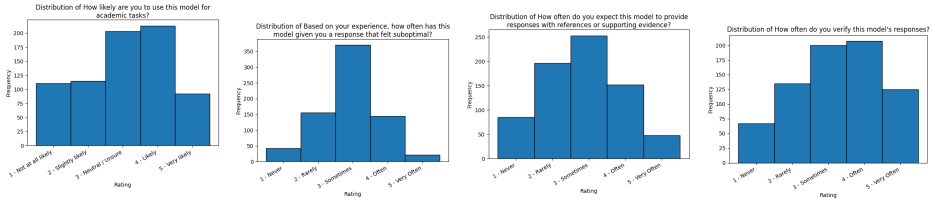

Figure 1: Distributions of numerical features.

2.2 Data issues

- The dataset is missing at least 118 values, e.g. student_id = 3. Missing numerical features were imputed as 3 (neutral); see `build_features` in `preprocessing.py`.
- Two data points (student_id = 2 and 15) contained a nonsensical feature "#NAME?". These were removed prior to training to prevent the model from learning from it.

2.3 Preprocessing

- To preserve ordinal information, Likert-scale features were converted to numerical by extracting the first character using regex pattern matching; see `extract_rating` in `preprocessing.py`.

- Multi-select survey responses were converted into binary indicator features using MultiLabelBinarizer; see `build_features` in `preprocessing.py`.
- The text data in the dataset was preprocessed by applying TF-IDF vectorization to each free-text column, converting the raw text into numerical features that capture the importance of each word relative to the whole; see `preprocessing_tfidf.py`.

2.4 Data Splitting

- data-split.py groups the cleaned dataset by student_id and randomly splits 85/15 for training/test. The file training.csv was used for training and hyperparameter tuning; test.csv was only used to measure final model performance.

3 Methodology

3.1 Logistic Regression

Logistic (softmax) regression is a stable and efficient classifier for simple datasets such as this. It readily affords further techniques such as feature mapping to capture nonlinear relationships and regularization to prevent overfitting our small training set. We tuned a logistic regression model both including and excluding text features represented by TF-IDF, but observed that including TF-IDF features improved validation performance at the cost of higher variance between folds.

Optimization: For training, we used the built-in optimizers in the `LogisticRegression` class from scikit-learn. We chose the `lbfgs` solver for L2 regularization and `saga` solver for L1 regularization. We set `max_iter=5000` to ensure convergence when tuning hyperparameters, but used `max_iter=1000` for final testing to meet the computational requirements.

Hyperparameters: The optimal combination of hyperparameters was selected by a grid search; see `log_reg_tuning` and `log_reg_tfidf_tuning`. The search range for each hyperparameter is justified below.

- `max_features`: the maximum number of tokens considered in the TF-IDF representation of text features. We searched over the range $\{100, 300, 500, 1000\}$, and validation results (see Figure 2) showed that accuracy, precision, recall, and F1 score are all maximized at `max_features` $= 300$, providing evidence that this range is reasonable for our dataset.
- `n-gram range`. Allows the TF-IDF representation to consider short phrase combinations, rather than just individual words. We tried introducing bigrams (two word combinations), searching over the range `n-gram range` $\in \{(1,1),(1,2)\}$. However we noticed immediate overfitting as n increased as the mean training accuracy was 0.993 and the mean validation accuracy was 0.674 for $(1,2)$, as opposed to mean training accuracy 0.854 and mean validation accuracy 0.688 for $(1,1)$.

- **Regularization strength:** $C \in \{0.1, 0.5, 1, 5, 10\}$, controlling the inverse of regularization. From Figure 2, we observe the trend of slightly underfitting when C is too small, thus regularization is too strong, and overfitting when C is too large.
- **Penalty:** We experimented with both L1 and L2 regularization. L2 regularization encourages smaller coefficients and generally yields stable models, while L1 can produce sparse coefficients, potentially useful if some features are irrelevant. We could have also tried elasticnet which combines L1 and L2, but we found that L2 significantly outperformed L1.
- **Solver:** We used `lbfgs` for L2 regularization and `saga` for L1 regularization, as recommended by the `scikit-learn` documentation [2].

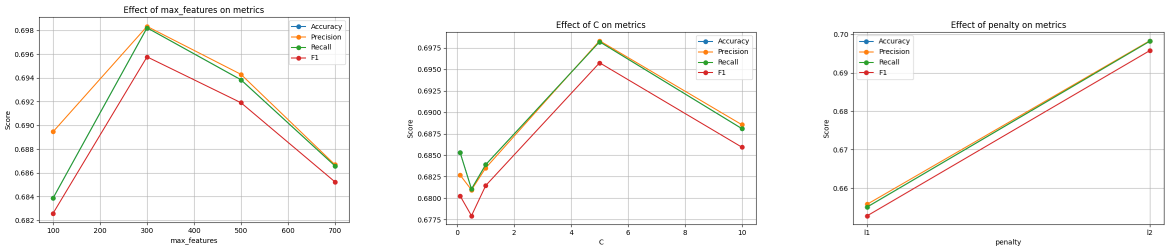

Figure 2: Logistic regression validation results as hyperparameters vary.

3.2 Naive Bayes Classifier

Naive Bayes is well-suited for our dataset because most survey features are discrete, low-cardinality variables such as Likert-scale responses and binary indicators. The naive Bayes assumption may not be perfectly true, for example the numerical features regarding likelihood of academic use and frequency of suboptimal responses are plausibly correlated. However, it seems reasonable to assume that features are approximately conditionally independent given class, hence a naive Bayes classifier can be very effective.

Optimization: Naive Bayes does not require iterative optimization. Parameters have closed-form maximum-likelihood estimates derived from empirical counts (categorical features) or means/variances (numerical features).

Hyperparameters: We tuned the Laplace smoothing coefficient α, which addresses the zero-probability problem in Bayesian classifiers by adding a small constant α to each feature count. The search range for α, namely $\alpha \in \{0.01, 0.1, 0.5, 1.0, 2.0, 5.0, 10.0\}$, was chosen following the recommendation of Manning, Raghavan, and Schütze [1]. The effect of α on the evaluation metrics is shown in Figure 3.

3.3 Random Forest Classifier (1 page)

Decision trees are suitable for mixed feature types and capturing relationships between numerical features and binary categorical features, which form an important part of our dataset. The numerical features exhibit relatively low entropy distributions, with noticeable concentration near rating 3 to 4. This means sampled values are more predictable,

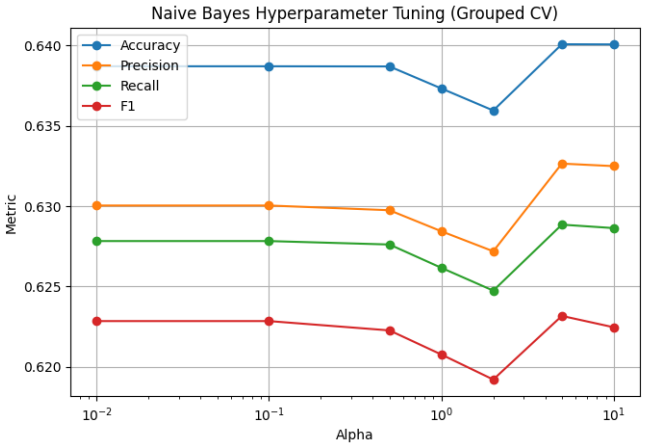

Figure 3: Naive Bayes validation results as the Laplace smoothing coefficient varies.

and information gain from numerical feature splits can be quite high. We can extend this to random forests, using ensemble averaging to avoid overfitting.

3.4 Validation

We employed grouped 5-fold cross-validation using `GroupKFold`, grouping by `student_id` to avoid data leakage across folds; see `logistic_regression_tfidf.py`.

3.5 Evaluation

We evaluated our models using accuracy, precision, recall, and F1. The final decision was heavily based on accuracy and F1, which combines precision and recall.

4 Results (1 page)

4.1 Logistic Regression

Table 1 summarizes the best combination of hyperparameters for logistic regression. Table 2 contains the resulting mean evaluation metrics of the test performance.

Model	Penalty	Solver	C	Max Features	N-grams
TF-IDF	L2	saga	5	300	(1,1)
No Text	L1	saga	0.1	N/A	N/A

Table 1: Logistic regression hyperparameters with strongest 5-fold CV performance.

Error Analysis: As evidenced by the confusion matrix (see Figure 4), the model achieved strong predictive performance (accuracy: 0.82, F1: 0.85) of ChatGPT, which it predicted more frequently, but often confused Claude (F1: 0.56) and Gemini (F1: 0.55). The top

two highest-confidence misclassifications predicted ChatGPT when the true labels were Claude (confidence: 0.998490) and Gemini (confidence: 0.986409).

4.2 Naive Bayes Classifier

The optimal Laplace smoothing coefficient was $\alpha = 5.0$. The test performance of this model is summarized in Table 2.

4.3 Random Forest Classifier

4.4 Model Comparison

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression (no text)	0.650	0.643	0.650	0.645
Logistic Regression (with TF-IDF)	0.659	0.653	0.659	0.655
Naive Bayes	?	?	?	?
Random Forest	?	?	?	?

Table 2: Evaluation metrics for the three model families on the test set.

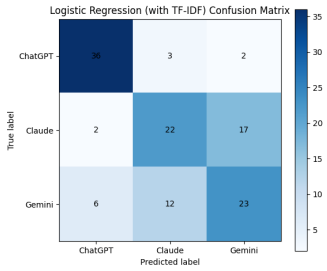

Figure 4: Confusion matrices for logistic regression, naive Bayes, and random forest models.

The model families from strongest to weakest performance were: [RANK MODELS]. The estimated test performance of [BEST MODEL + 1-NUMBER ACCURACY ESTIMATE]. This estimate is the average accuracy across 5-folds CV and the test performance, which were all quite consistent.

5 Contributions and Learning

- Roy Gal:
- William Gao: My main contribution was formulating the project methodology and writing the proposal and final report. The most important lesson I learned is to use a variety of evaluation metrics when selecting a final model.
- Zehao Peng:
- Steven Qiao:

This section should contain a list of 3-4 bullets, one for each student. Each student should write at most three sentences describing their main contributions and the most important lesson learned.

References

[1] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[2] scikit learn. sklearn.linear_model.logisticregression — scikit-learn 1.6.2 documentation. `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`, 2025.