# Evaluation Report: Developer Role Prediction from Commit Records

Adith N Reganti-IIT2024249

## 1 Introduction

This project explored whether it is possible to predict the role of a developer (frontend, backend, fullstack, QA) from individual commit records. Each record contained a commit message, categorical details (commit type, file extensions, time of commit), and numeric signals (lines added/deleted, files changed, comments). The focus was not only on reaching high accuracy but also on reasoning carefully about preprocessing, feature choices, and model behavior.

## 2 Exploratory Data Analysis (EDA)

We began by checking the quality of the dataset. There were no missing values or duplicates, so the data was structurally sound. The target distribution showed four roles with some imbalance but nothing severe.

### Key Findings and Visualizations

- **File Extensions:** After cleaning the raw values (removing `np.str_`), we created a cross-tabulation of extensions against roles. A **heatmap** showed strong associations: `.css` and `.html` linked to frontend, `.py` and `.sql` to backend, test/config files (`test_py`, `yml_json`) to QA, while `.js_ts` overlapped between frontend and fullstack.
- **Numeric Features:** For lines added, lines deleted, comments, and files changed, we plotted **histograms** with different bin sizes to inspect skewness. **Boxplots by role** revealed that QA commits are generally the smallest, backend commits moderate, and fullstack the largest. **Pairplots** of numeric features showed overlap between roles but highlighted correlations such as lines added vs. lines deleted.
- **Engineered Features:** We introduced features like *net churn*, *add/delete ratio*, and *files-per-line*. **Pairplots** and a **correlation heatmap** were used to verify that these captured commit style (e.g., QA editing many files lightly, backend editing fewer files more deeply).
- **Temporal Features:** We binned hours into "night", "morning", "afternoon", and "evening". **Countplots** showed that QA commits skewed toward evening/night, while backend activity concentrated in daytime hours. Weekly distribution plots confirmed all roles were most active on weekdays.
- **Commit Type:** A **countplot** of commit type vs. role showed QA strongly aligned with "test", while backend and frontend dominated "feature" and "bugfix".

Overall, these visualizations confirmed that commit messages and file extensions are the strongest predictors, while numeric and temporal features provide useful context and interpretability.

## 3 Models and Results

We compared three models:

- **Gaussian Naive Bayes:** A baseline that assumes numeric features follow normal distributions. Performance was weak (Macro F1 $\approx 0.63$) because text features violated these assumptions.
- **Multinomial Naive Bayes:** A natural fit for Bag-of-Words commit messages. It captured strong word-role patterns (e.g., "fix test" for QA) and achieved Macro F1 $\approx 0.97$.
- **Random Forest:** An ensemble that integrates numeric, categorical, and text features. It reduced errors where roles overlapped and achieved the best performance (Macro F1 $\approx$ 0.98).

| Model | Macro F1 | Accuracy | Notes |
|---|---|---|---|
| GaussianNB | 0.63 | 62% | Weak baseline, text not well modeled |
| MultinomialNB | 0.97 | 97% | Very strong, text dominates |
| Random Forest | 0.98 | 98% | Best overall, balanced across roles |

## 4 Error Analysis

Most errors came from confusion between frontend and fullstack, which makes sense since both touch similar files (`.js_ts`, `.html`). Commit messages alone achieved very high accuracy, showing that language dominates the signal, but this also raises the risk of overfitting to wording styles. Numeric and categorical features acted as a safety net, especially for commits with short or vague messages.

## 5 Lessons Learned

- Text features are extremely powerful, but can cause over-reliance.
- Feature engineering (e.g., churn ratios) improves interpretability and helps explain model behavior.
- A strong baseline (Naive Bayes) is important before moving to complex models.

## 6 Conclusion

Our final Random Forest model reached 98% accuracy and a Macro F1 of 0.98. The model was reliable across all roles, but the evaluation showed that frontend and fullstack remain challenging to separate. We could further try to improve this by using other NLP techniques like Word-2-Vec or even use Sentence Transformers to capture the semantic meaning of the commit messages.