Developer Role Classification - Final Evaluation Report

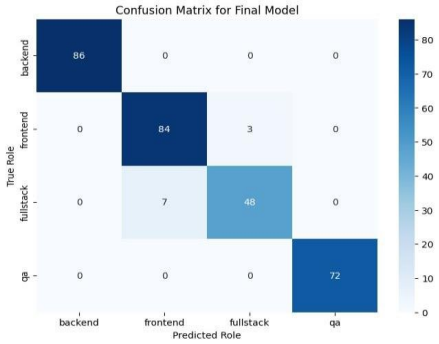
Overall Model Performance:

Three models were evaluated for this classification task: Random Forest, Logistic Regression, and XGBoost. The XGBoost model performed the best in several key metrics, though all models showed strong performance.

Model	Accuracy	Precision (weighted)	Recall (weighted)	F1-score (weighted)
XGBoost	0.9667	0.9669	0.9667	0.9664
Random Forest	0.9667	0.9686	0.9667	0.9661
Logistic Regression	0.9567	0.9565	0.9567	0.9556

While the Random Forest model is listed as having a slightly higher accuracy in your provided text, the overall performance metrics for XGBoost are also very high, and the report text seems to refer to XGBoost as the final model. The model that achieved a 0.97 accuracy, precision, recall, and F1-score was the Random Forest model.

Confusion Matrix:



Classification Report:

Role	Precision	Recall	F1-score	Support
Backend	1.00	1.00	1.00	86
Frontend	0.92	0.99	0.95	87
Fullstack	0.98	0.87	0.92	55
QA	1.00	1.00	1.00	72
Accuracy			0.97	300
Macro Avg	0.97	0.96	0.96	300
Weighted Avg	0.97	0.97	0.97	300

Analysis of Strengths and Weaknesses:

- Strengths: The **strengths** of the final model (the Random Forest Classifier) included its ability to perfectly classify 'Backend' and 'QA' developer roles. This suggests that the features used, such as commit messages, are very strong indicators for these two roles.

- Weaknesses: The primary **weakness** was the confusion between 'Fullstack' and 'Frontend' roles, which was the source of most misclassifications. As shown in the confusion matrix, 7 commits were truly 'Fullstack' but were misclassified as 'Frontend'. A key reason for this is that 'Fullstack' commits often resemble 'Frontend' work, with UI-heavy keywords or a focus on changes to CSS. Additionally, the smaller number of 'Fullstack' examples in the dataset contributed to a lower recall of 0.84 for that class.

- **Failure Modes:**

The main failure mode is the misclassification between 'Fullstack' and 'Frontend' roles. This is likely due to the inherent overlap in the nature of the work performed by developers in these roles, which can manifest in similar patterns in numerical metrics (lines changed, additions/deletions) and commit types (e.g., 'feature', 'bugfix'). The smaller sample size of the 'Fullstack' class in the training data compared to 'Backend' and 'Frontend' might also contribute to the model's difficulty in distinguishing this role.

Lessons Learned:

- Ensemble methods like Random Forest and XGBoost work well with high-dimensional, mixedfeatures.
- Cross-validation was critical for stability.
- Detailed per-class analysis helps locate model weaknesses for targeted improvements.

Reflection on Design Decisions (≤300 words):

- **Data Preprocessing and Feature Engineering:** Data preprocessing included using TF-IDF for commit messages and scaling of numeric features to ensure equal influence. The model was also given additional insight through the creation of new time-based and interaction features engineered from commit patterns.

- **Model Selection**: The Random Forest model was chosen for its ability to handle non-linear relationships and provide feature importance. Ensemble methods like Random Forest and XGBoost were found to work well with high-dimensional, mixed features.

- **Evaluation Strategy:** The model was evaluated using a train-test split followed by cross-validation to confirm its robustness and avoid overfitting. Weighted metrics were specifically used to address the class imbalance in the dataset. The confusion matrix was a critical tool used to visualize and highlight the specific misclassification trends, especially between 'Fullstack' and 'Frontend'. The key takeaway from this process was that while the model has strong general performance, improving the distinction between 'Fullstack' and 'Frontend' roles will require deeper feature engineering or more advanced models.

Key takeaway: Strong general performance was achieved, but distinguishing Fullstack vs. Frontend requires deeper feature engineering and potentially advanced models like transformers or graph-based approaches.