Evaluation Report: Developer Role Classification Model

This report summarizes the final evaluation, major failure modes, and key lessons learned from the project to classify a developer's role based on their Git commit data. The project explored various machine learning approaches, with a significant focus on feature engineering.

Final Evaluation

The project evaluated several models, with performance progressively improving as more advanced feature engineering and modeling techniques were applied. The final performance metrics (Accuracy and Macro F1 Score) for the key models are summarized below:

Model	Text Embedding Method	Accuracy	Macro F1 Score
Logistic Regression (Baseline)	TF-IDF	0.940	0.931
Decision Tree (Untuned)	TF-IDF	0.940	0.931
Decision Tree (Tuned)	TF-IDF	0.967	0.964
Random Forest	**TF-IDF**	**0.973**	**0.969**

The **Random Forest classifier** achieved the highest performance with an accuracy of **97.3%**. The significant improvement of the tuned Decision Tree (from 94.0% to 96.7%) and the even better performance of the Random Forest highlight the importance of both hyperparameter optimization and the use of ensemble methods for this particular dataset.

Major Failure Modes

1. **Initial Underperformance of the Decision Tree:** A notable initial failure was the fact that the untuned Decision Tree performed no better than the simple Logistic Regression baseline. Both models achieved a 94% accuracy, which was unexpected given that decision trees can typically capture more complex, non-linear relationships. This indicated that the default parameters of the Decision Tree were not well-suited for this dataset and that hyperparameter tuning was essential. Without tuning, the model was unable to leverage its potential advantages.

2. **Lack of Advanced Text Embeddings:** While the TF-IDF vectorization proved effective, it represents a more traditional approach to text analysis. A potential failure mode lies in not exploring more advanced embedding techniques like Word2Vec, GloVe, or transformer-based models (e.g., BERT). These methods can capture semantic relationships and context in the commit messages, which TF-IDF, being a frequency-based measure, cannot. This could have potentially unlocked even higher performance, especially in distinguishing between nuanced commit messages.

Lessons Learned

1. **Hyperparameter Tuning is Crucial:** The most significant lesson learned was the dramatic impact of hyperparameter tuning. The Decision Tree's accuracy jumped by nearly 3% after a thorough GridSearchCV. This demonstrates that default model parameters are often suboptimal and that a systematic search for the best parameters is a critical step in the machine learning workflow to maximize a model's predictive power.
2. **Ensemble Methods Provide Robustness:** The success of the Random Forest classifier over the tuned single Decision Tree highlighted the value of ensemble learning. The author correctly identified that the problem was likely one of high variance (individual trees overfitting) rather than bias. By using bagging (Bootstrap Aggregating), the Random Forest averaged out the errors of multiple decision trees, leading to a more stable and accurate model. This reinforces that for problems where individual models might be sensitive to the training data, an ensemble approach is often superior.
3. **EDA Informs Feature Importance:** The exploratory data analysis (EDA) was vital for justifying the inclusion of various features. The heatmaps and bar plots clearly showed which features, such as committype and linesadded, were strong differentiators for developer roles. For instance, the observation that test commits were almost exclusively from QA developers and that fullstack developers added the most lines of code provided a strong rationale for their use in the model. This underscores the importance of visualizing data to confirm feature relevance before modeling.

Reflection on Design Decisions

My most important design decision was to **employ GridSearchCV to systematically tune the hyperparameters of the Decision Tree classifier**. After observing that the untuned Decision Tree performed no better than the Logistic Regression baseline, I knew that simply moving on to another model would be a missed opportunity. I justified this decision by recognizing that the model's default settings were likely not optimal for the specific patterns within the Git commit dataset. The search space I defined for the grid search was comprehensive, covering different splitting criteria, tree depths, and feature consideration strategies. This methodical approach was validated when the tuned model's accuracy significantly increased to 96.7%, outperforming the baseline and proving that the model was indeed capable of capturing the underlying patterns when properly configured.

Another critical design decision was to **progress from a single Decision Tree to a Random Forest classifier**. I reasoned that even with tuning, a single tree might be prone to overfitting

and sensitive to small variations in the training data—a problem of high variance. A