Random Forest Model

The Random Forest model was evaluated using a `param_grid` with 48 combinations of hyperparameters.

Baseline Performance

Runlogs

The baseline Random Forest model, without any oversampling or undersampling techniques, achieved the following top performance:

- **Accuracy:** 0.6712 (Run 11)
- **Precision:** 0.4842 (Run 11)
- **Recall:** 0.5874 (Run 34)
- **F1-score:** 0.5293 (Run 34)
- **ROC_AUC:** 0.7182 (Run 47)

The model generally showed moderate performance, with a relatively low precision indicating a notable number of false positives.

SMOTE Performance

Runlogs

When SMOTE was applied for oversampling, the Random Forest model's performance changed as follows:

- **Accuracy:** Range 0.61 to 0.64, with the highest at 0.6442 (Run 11)
- **Precision:** Range 0.43 to 0.46, with the highest at 0.4632 (Run 26)
- **Recall:** Significantly increased, ranging from 0.70 to 0.78, with the highest at 0.7822 (Run 16)
- **F1-score:** Range 0.54 to 0.56, with the highest at 0.5656 (Run 47)
- **ROC_AUC:** Range 0.71 to 0.72, with the highest at 0.7295 (Run 4)

SMOTE effectively increased recall, which is beneficial for identifying more positive cases. However, this came at the cost of reduced precision and overall accuracy compared to the baseline. The F1-score and ROC_AUC showed a slight improvement or similar performance, suggesting a better balance between precision and recall, but still not a significant leap.

SMOTE+Tomek Performance

Runlogs

Combining SMOTE with Tomek links for undersampling yielded these results:

- **Accuracy:** Range 0.61 to 0.64, with the highest at 0.6445 (Run 6)
- **Precision:** Range 0.43 to 0.46, with the highest at 0.4609 (Run 6)
- **Recall:** High, ranging from 0.71 to 0.77, with the highest at 0.7794 (Run 21)
- **F1-score:** Range 0.54 to 0.56, with the highest at 0.5652 (Run 34)
- **ROC_AUC:** Range 0.71 to 0.72, with the highest at 0.7252 (Run 46)

SMOTE+Tomek showed similar trends to SMOTE alone, maintaining a high recall but still exhibiting lower precision and accuracy compared to the baseline. The F1-score and ROC_AUC remained in a similar range.

XGBoost Model Analysis

The XGBoost model was evaluated across various hyperparameters including learning rate, max depth, number of estimators, and subsample ratio.

Baseline Performance

Runlogs

The baseline XGBoost model demonstrated the following top performance:

- **Accuracy:** 0.6754 (Run 28)
- **Precision:** 0.4897 (Run 28)
- **Recall:** 0.6016 (Run 6)
- **F1-score:** 0.5373 (Run 6)
- **ROC_AUC:** 0.7260 (Run 28)

XGBoost baseline generally outperformed the Random Forest baseline in terms of accuracy, precision, F1-score, and ROC_AUC, indicating a stronger initial performance.

SMOTE Performance

Runlogs

When SMOTE was applied to the XGBoost model:

- **Accuracy:** Range 0.61 to 0.65, with the highest at 0.6548 (Run 28)
- **Precision:** Range 0.43 to 0.47, with the highest at 0.4703 (Run 28)
- **Recall:** Significantly increased, ranging from 0.66 to 0.78, with the highest at 0.7819 (Run 1)
- **F1-score:** Range 0.53 to 0.56, with the highest at 0.5689 (Run 8)
- **ROC_AUC:** Range 0.70 to 0.73, with the highest at 0.7307 (Run 28)

Similar to the Random Forest model, SMOTE notably boosted recall for XGBoost, while precision and accuracy saw a decrease compared to the baseline. The F1-score and ROC_AUC showed modest improvements in some cases, but generally remained within a similar range as the non-SMOTE version, albeit with a different precision-recall balance.

SMOTE+Tomek Performance

Runlogs

The combination of SMOTE+Tomek with XGBoost showed the following top performance:

- **Accuracy:** 0.6712 (Run 11)
- **Precision:** 0.4842 (Run 11)
- **Recall:** 0.5874 (Run 34)
- **F1-score:** 0.5293 (Run 34)
- **ROC_AUC:** 0.7182 (Run 47)

Based on the available data, the initial trend for SMOTE+Tomek with XGBoost appears to be similar to SMOTE, aiming to improve recall at the expense of other metrics.

Conclusion

From the analysis of both Random Forest and XGBoost models across different sampling techniques, the following conclusions can be drawn:

1. **Baseline Performance:** XGBoost generally performed better than Random Forest in its baseline configuration, achieving higher accuracy, precision, F1-score, and ROC_AUC.

This suggests that XGBoost is a more suitable model for this particular risk assessment task when no sampling is applied.

2. **Impact of Oversampling (SMOTE):** For both models, applying SMOTE significantly improved recall. This is crucial for risk assessment scenarios where correctly identifying positive cases (e.g., risky code merges) is paramount, even if it leads to an increase in false positives (lower precision). However, this improvement in recall came at the cost of overall accuracy and often precision. The F1-score and ROC_AUC, which consider both precision and recall, showed only minor improvements.
3. **Impact of Combined Sampling (SMOTE+Tomek):** The combination of SMOTE and Tomek links yielded very similar results to SMOTE alone for both models. While the intention of Tomek links is to clean up the dataset after oversampling, its impact on the overall performance metrics (accuracy, precision, F1-score, ROC_AUC) was not significantly different from using SMOTE only.
4. **Overall Efficacy of Sampling:** The description states, "From all these there experiment we can see that SMOTE random forest model performing some what decent but not significantly enough." This sentiment extends to the XGBoost model as well. While SMOTE and SMOTE+Tomek are effective in balancing the dataset and improving recall, the overall impact on the F1-score and ROC_AUC (which are often key metrics for imbalanced classification) is not a "significant" leap forward.

In summary, for an AI-powered risk assessment system aiming to identify code merges, **XGBoost appears to be the better performing model overall in its baseline configuration.** While oversampling techniques like SMOTE and SMOTE+Tomek successfully increase recall, their benefits in terms of a comprehensive performance improvement (as measured by F1-score and ROC_AUC) are modest. Therefore, further investigation into hyperparameter tuning, feature engineering, or exploring other advanced ensemble methods with more sophisticated balancing strategies might be necessary to achieve a "significantly enough" performance improvement.