Data Preparation

I began by loading the dataset from a CSV file and immediately examined its overall shape and structure to understand the volume and types of data available. Duplicate rows were identified and removed to prevent redundancy from skewing the analysis. Specific placeholder characters I identified (like the “?” and “*” characters) were replaced with NaN values, and these missing values were then replaced with the mode of each column. This approach helped maintain the natural distribution of the data. In addition, I generated plots for each column to visually inspect the distribution of values, which confirmed that the features were fairly balanced. Finally, I split the dataset into predictors and the target variable and applied one-hot encoding (dropping the first category) to the categorical features to prepare the data for model training.

Insights from Data Preparation

The initial inspection revealed several issues that needed addressing. Duplicate records were present and were appropriately removed, and missing values were identified in multiple columns. By replacing these missing values with the mode, I was able to preserve the original distribution of the data. The visualizations further confirmed that the features did not exhibit significant imbalances, providing confidence that the cleaned dataset was prepared for the subsequent modeling phase. After looking over these figures, I was ready to move forward to training the models and comparing their performance.

Model Training Procedure

I split the encoded dataset into training and test sets using stratified sampling to ensure that the class proportions remained consistent across both sets. First, I built a baseline K-Nearest Neighbors (KNN) classifier using default parameters. This initial model achieved a precision of approximately 66% for the majority class (no-recurrence-events) but only around 20% for the minority class (recurrence-events), with an overall accuracy

near 60%. To improve the performance, I employed GridSearchCV to tune the KNN model by testing a range of neighbor values from 1 to 10 with 5-fold cross-validation. This tuning process identified 10 neighbors as the optimal parameter, resulting in a boosted performance for the majority class—with precision increasing to about 69% and recall reaching as high as 94%—although the recall for the minority class remained low (approximately 8%). In parallel, I trained a Logistic Regression model, setting the maximum iterations to 1,000 to ensure convergence. This model achieved around 70% precision for the majority class and about 45% for the minority class, with an overall accuracy of roughly 67%.

Model Performance and Confidence

The evaluation of each model was based on standard classification metrics. The baseline KNN model and its tuned variant both demonstrated a strong ability to predict the majority class correctly, as evidenced by precision values in the mid-60s to high 60s. However, the low recall for the minority class in both cases suggests that these models struggled to identify recurrence events effectively. The Logistic Regression model, while providing a more balanced performance, still revealed challenges in classifying the less frequent class. The combination of cross validation and hyperparameter tuning techniques used in this study gives me confidence that the models generalize well to unseen data. However, I do think that the performance metrics indicate that further improvements like experimenting with more complex models or applying techniques like class weighting or oversampling could enhance the detection of the minority class even further.

Conclusion

Overall, the models all showed high promise despite a relatively simple approach to training. The fidelity of the approach was bolstered by ensuring all the training, test, and validation data was properly prepared and maintained throughout the process.