### **Model Evaluation Report**

[Model Evaluation Report 1](#_Toc610146527)

[Introduction 1](#_Toc860954976)

[Classification Report Breakdown 1](#_Toc1655737301)

[Overall Performance 2](#_Toc974679252)

[Key Observations 2](#_Toc1044863745)

[Recommendations for Improvement 3](#_Toc211830228)

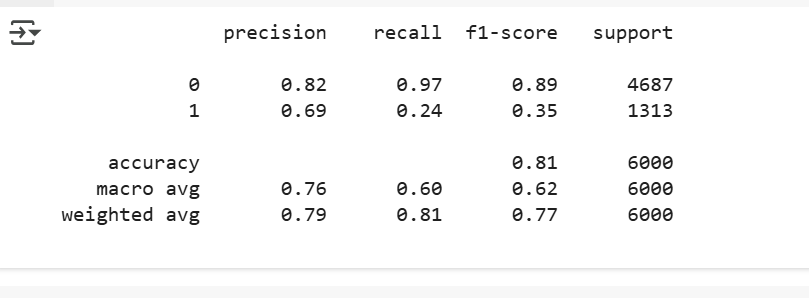
[Conclusion 3](#_Toc772462382)

#### **Introduction**

This report provides an evaluation of the performance of the classification model. The metrics considered include **Precision**, **Recall**, **F1-score**, and **Support** for each class, along with the overall accuracy of the model. The evaluation is based on the confusion matrix generated from the predicted and actual values in the test set.

#### **Classification Report Breakdown**

The classification report presents key performance metrics for each class in the dataset. Here is a summary of these metrics:



**Precision**:

* Precision for class 0 is **0.82**, meaning that of all the instances predicted as class 0, 82% were correct.
* Precision for class 1 is **0.69**, meaning that of all the instances predicted as class 1, 69% were correct.

**Recall**:

* Recall for class 0 is **0.97**, indicating that 97% of actual class 0 instances were correctly identified by the model.
* Recall for class 1 is **0.24**, indicating that only 24% of actual class 1 instances were correctly identified by the model.

**F1-score**:

* The F1-score for class 0 is **0.89**, representing a good balance between precision and recall.
* The F1-score for class 1 is **0.35**, which is lower due to the low recall for class 1.

**Support**:

* The support represents the number of actual instances of each class. For class 0, there are **4687** instances, while for class 1, there are **1313** instances.

#### **Overall Performance**

* **Accuracy**: The model has an overall accuracy of **81%**, meaning it correctly predicted 81% of the test set instances.

**Macro Average** (unweighted average of precision, recall, and F1-score across both classes):

* **Precision**: **0.76**
* **Recall**: **0.60**
* **F1-score**: **0.62**

**Weighted Average** (accounts for the class imbalance):

* **Precision**: **0.79**
* **Recall**: **0.81**
* **F1-score**: **0.77**

#### **Key Observations**

* **Class Imbalance**: There is a notable imbalance between the two classes: **Class 0** (4687 instances) and **Class 1** (1313 instances). This imbalance is likely the reason why the model performs well in predicting the majority class (class 0) but struggles to identify the minority class (class 1).
* **Recall for Class 1**: The recall for class 1 is particularly low (**0.24**), which suggests that the model is not effectively identifying instances of the minority class. This could be due to the class imbalance, as the model tends to favor the majority class in such scenarios.

#### **Recommendations for Improvement**

1. **Adjust the Decision Threshold**: The model's decision threshold can be adjusted to make it more sensitive to the minority class. This can help improve recall for class 1, although it may reduce precision.
2. **Address Class Imbalance**:
   1. **SMOTE (Synthetic Minority Over-sampling Technique)**: SMOTE can be used to oversample the minority class and help balance the dataset.
   2. **Class Weights**: Consider using models that allow setting class weights (e.g., LogisticRegression with the class\_weight='balanced' parameter) to penalize misclassifications of the minority class more heavily.
3. **Model Exploration**: Given the imbalance, it might be beneficial to explore more advanced models or ensemble techniques:
   1. **Random Forest**: Random forests handle imbalanced datasets well due to their ability to randomly sample and use multiple decision trees.
   2. **XGBoost**: XGBoost has built-in mechanisms to deal with class imbalance and may improve performance on the minority class.

#### **Conclusion**

The current model shows promising overall performance with an accuracy of 81%. However, the imbalance between classes significantly affects the recall for the minority class (class 1). Addressing this imbalance through techniques like resampling, adjusting decision thresholds, or exploring other models can improve performance, especially for detecting instances of class 1.