Ice Classification in the Groenland

Alaa Bouattour - Mahdi Ben Ayed
M2QF

November 17, 2024

Abstract

This report details the methodology, experiments, and outcomes of a machine learning lab focused on supervised learning algorithms for binary classification. Key steps include data preprocessing, feature engineering, and model evaluation to predict ice presence using infrasound signals and meteorological data. The results highlight the performance of various classifiers and their applicability to the task.

1 Methodology

1.1 Data Preprocessing

- **Threshold Selection:** The target variable $Y1$ represents infrasound signal intensity and contains values that are either 0 or greater than 0. To transform this quantitative variable into a binary classification problem, we define the threshold as 0. Values of $Y1 > 0$ are classified as the positive class (*high ice presence*), while values of $Y1 = 0$ are classified as the negative class (*low/no ice presence*). This choice of threshold ensures a clear separation between the two classes based on the physical interpretation of the signal.

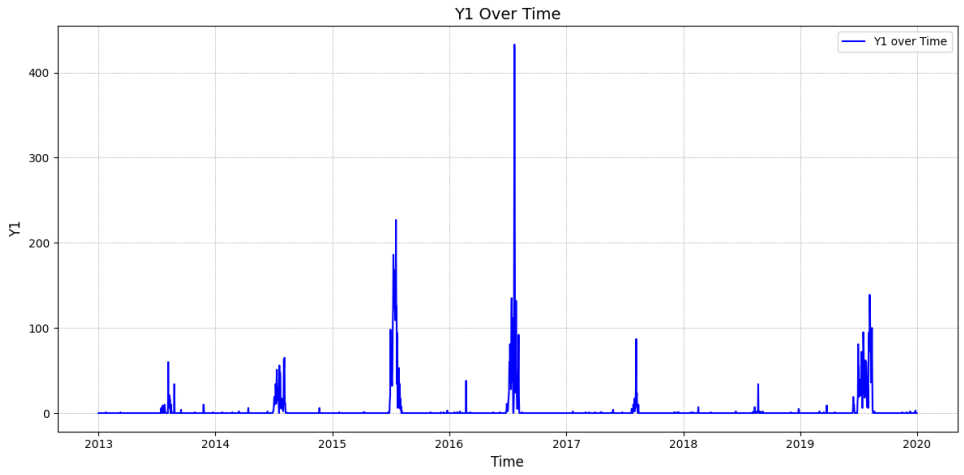

Figure 1: $Y1$ over time

- **Feature Engineering:** The feature engineering process involved converting the time column to a datetime object, enabling the extraction of month, day, and year

as separate features. The target variable Y1 was transformed into a binary classification target (0 or 1) for compatibility with classification models.
Additionally, exploratory analysis visualized the distribution of non-zero target values across months, highlighting potential temporal patterns for model insights. The monthly distribution of non-zero $Y1$ events, as shown in Figure 2, highlights the significant role of the `month` feature in predicting ice presence. A clear seasonal pattern emerges, with the majority of non-zero $Y1$ events occurring in the summer months (June, July, and August). This suggests that the `month` feature provides valuable temporal information for the classification task.

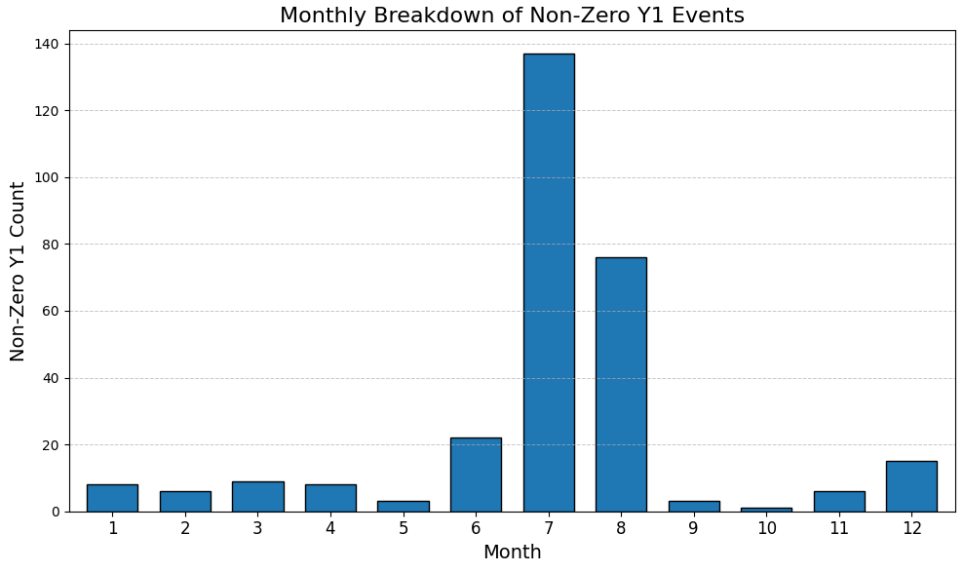

Figure 2: Monthly Breakdown of Non-Zero $Y1$ Events

- **Train-test split:** The dataset was split into training, validation, and test sets to ensure proper evaluation and avoid data leakage. The training set was used to train the model, while the validation set was used for hyperparameter tuning and determining the optimal decision threshold. The test set remained unseen during training and validation to provide an unbiased assessment of the model's performance. This approach ensures robust evaluation and minimizes overfitting, especially important for imbalanced datasets.

2 Algorithms

2.1 Decision Tree Classifier

The Decision Tree classifier was implemented to predict ice presence based on the given features. The model underwent extensive hyperparameter tuning using grid search with cross validation. The parameter grid included the following:

- `max_depth`: [3, 5, 10, None]
- `min_samples_split`: [2, 10, 20]
- `min_samples_leaf`: [1, 5, 10]

- `criterion`: ['gini', 'entropy']

To optimize the decision threshold, the precision-recall curve was calculated on the validation dataset, and the threshold corresponding to the maximum F1 score was selected. The best model parameters and the optimal threshold were then used to evaluate the performance on the test dataset.

Test Results

The evaluation of the Decision Tree model on the test dataset yielded the following metrics:

Table 1: Performance Metrics for the Decision Tree Classifier

ROC-AUC Score	Precision	Recall (Sensitivity)	F1-Score
0.8106	0.6842	0.6610	0.6724

The confusion matrix is illustrated in Figure 3, which visually represents the classifier's performance.

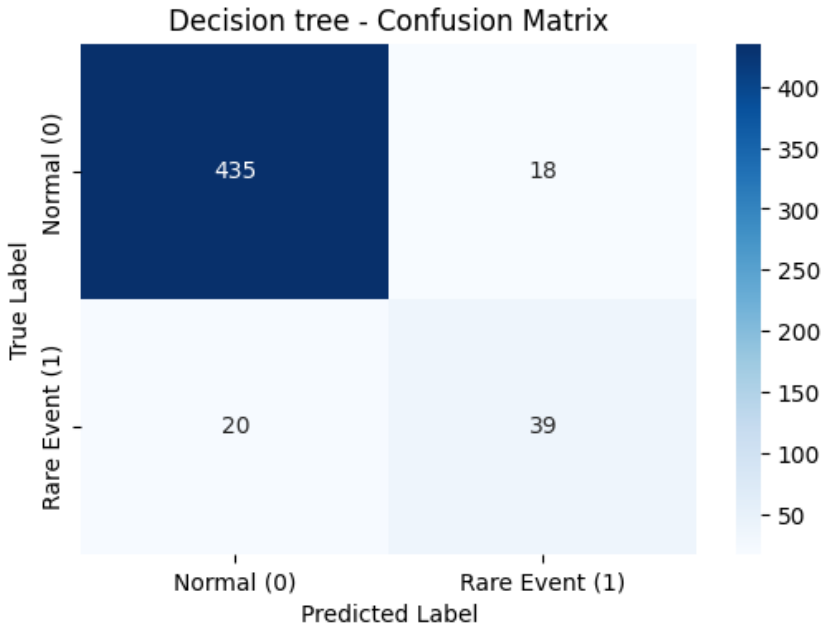

Figure 3: Confusion Matrix for the Decision Tree Classifier

Discussion

The Decision Tree classifier demonstrated reasonable performance with a balanced trade-off between precision and recall. The F1-Score of 0.6724 indicates a decent ability to balance false positives and false negatives.

2.2 Weighted Decision Tree Classifier

Figure 4: Proportion of Target Classes

To address the class imbalance in the dataset, a weighted Decision Tree classifier was implemented. The model utilized the `class_weight='balanced'` parameter to assign weights inversely proportional to class frequencies. Similar to the unweighted model, hyperparameter tuning was conducted using a grid search with cross-validation. The parameter grid included the following:

- `max_depth`: [3, 5, 10, None]
- `min_samples_split`: [2, 10, 20]
- `min_samples_leaf`: [1, 5, 10]
- `criterion`: ['gini', 'entropy']

To optimize the decision threshold, the precision-recall curve was calculated on the validation dataset, and the threshold corresponding to the maximum F1 score was selected. The best model parameters and the optimal threshold were then used to evaluate the performance on the test dataset.

Test Results

The evaluation of the weighted Decision Tree model on the test dataset yielded the following metrics:

Table 2: Performance Metrics for the Weighted Decision Tree Classifier

ROC-AUC Score	Precision	Recall (Sensitivity)	F1-Score
0.8733	0.5632	0.8305	0.6712

The confusion matrix is illustrated in Figure 5, which visually represents the classifier's performance.

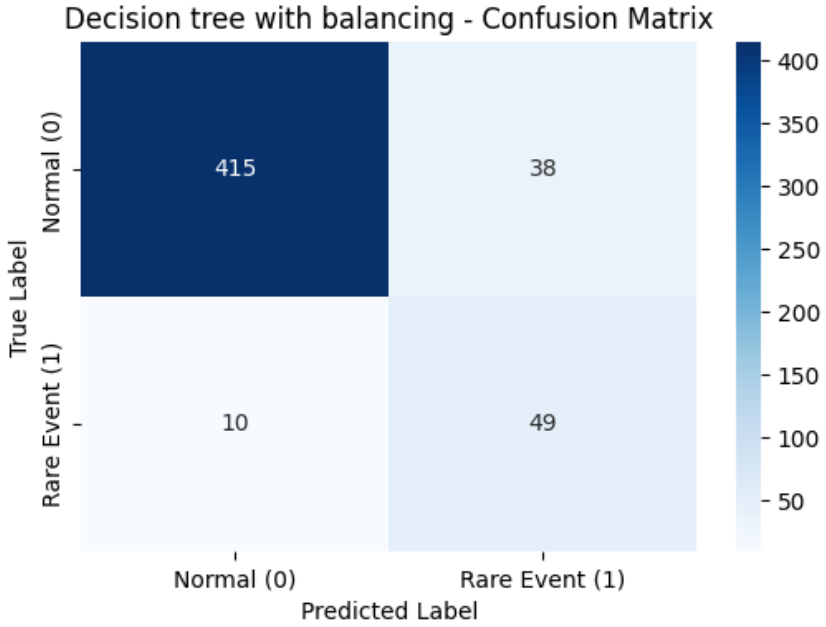

Figure 5: Confusion Matrix for the Weighted Decision Tree Classifier

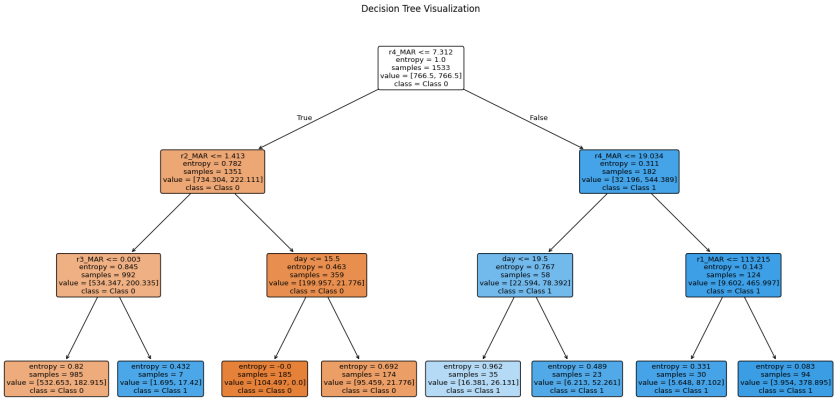

Figure 6: Weighted Decision Tree

Discussion

The weighted Decision Tree classifier demonstrated an improved ability to handle the class imbalance. The ROC-AUC score of 0.8733 indicates strong overall performance. Notably, the recall improved to 0.8305, capturing more true positives (49). However, this came at the expense of precision (0.5632), as the number of false positives increased (38). The F1-Score of 0.6712 reflects a balanced trade-off between precision and recall.

The Decision Tree was trained using `entropy` as the splitting criterion, balancing model complexity and interpretability. The most significant feature driving the decision at the root node was `r4_MAR`. The tree successfully handles the class imbalance in the dataset through careful hyperparameter tuning (`class_weight='balanced'`) and identifies meaningful patterns for minority class (Class 1) predictions. The final model exhibits man-

ageable depth and high interpretability, making it suitable for explaining classification decisions to stakeholders.

2.3 Random Forest Classifier

The model utilized `class_weight='balanced'` to address class imbalance by assigning weights inversely proportional to class frequencies. Hyperparameter tuning was performed using a grid search with cross-validation. The parameter grid included the following:

- `n_estimators`: [50, 100, 200]
- `max_depth`: [3, 5, 10, None]
- `min_samples_split`: [2, 10, 20]
- `min_samples_leaf`: [1, 5, 10]
- `criterion`: ['gini', 'entropy']

To optimize the decision threshold, the precision-recall curve was calculated on the validation dataset, and the threshold corresponding to the maximum F1 score was selected. The best model parameters and the optimal threshold were then used to evaluate the performance on the test dataset.

Test Results

The evaluation of the Random Forest model on the test dataset yielded the following metrics:

Table 3: Performance Metrics for the Random Forest Classifier

ROC-AUC Score	Precision	Recall (Sensitivity)	F1-Score
0.8394	0.7368	0.7119	0.7241

Discussion

The Random Forest classifier demonstrated strong performance with an F1-Score of 0.7241, balancing precision and recall effectively. Its ROC-AUC score of 0.8394 indicates good overall discriminative ability. Compared to the Decision Tree classifier, the Random Forest achieved higher precision (0.7368) and captured more true positives (42). The ensemble nature of the Random Forest likely contributed to its robustness by reducing variance and improving generalization.

2.4 XGBoost Classifier

The model was trained with a hyperparameter grid search, optimizing for the F1 score using cross-validation. The parameter grid included the following:

- `n_estimators`: [50, 100, 200]
- `max_depth`: [3, 5, 10]
- `learning_rate`: [0.01, 0.1, 0.2]
- `subsample`: [0.6, 0.8, 1.0]

- `colsample_bytree`: [0.6, 0.8, 1.0]
- `gamma`: [0, 1, 5]

To optimize the decision threshold, the precision-recall curve was calculated on the validation dataset, and the threshold corresponding to the maximum F1 score was selected. The best model parameters and the optimal threshold were then used to evaluate the performance on the test dataset.

Test Results

The evaluation of the XGBoost model on the test dataset yielded the following metrics:

Table 4: Performance Metrics for the XGBoost Classifier

ROC-AUC Score	Precision	Recall (Sensitivity)	F1-Score
0.8036	0.8043	0.6271	0.7048

Discussion

The XGBoost classifier demonstrated competitive performance, achieving a high precision of 0.8043, indicating a strong ability to minimize false positives. However, its recall (0.6271) was lower compared to other models, resulting in a moderate F1-Score of 0.7048. The ROC-AUC score of 0.8036 reflects good overall discriminative ability.

2.5 Extra Trees Classifier

The model incorporated `class_weight='balanced'` to address class imbalance by assigning weights inversely proportional to class frequencies. A grid search with cross-validation was conducted to optimize hyperparameters. The parameter grid included the following:

- `n_estimators`: [50, 100, 200]
- `max_depth`: [3, 5, 10, None]
- `min_samples_split`: [2, 10, 20]
- `min_samples_leaf`: [1, 5, 10]
- `criterion`: ['gini', 'entropy']

To optimize the decision threshold, the precision-recall curve was calculated on the validation dataset, and the threshold corresponding to the maximum F1 score was selected. The best model parameters and the optimal threshold were then used to evaluate the performance on the test dataset.

Test Results

The evaluation of the Extra Trees model on the test dataset yielded the following metrics:

Table 5: Performance Metrics for the Extra Trees Classifier

ROC-AUC Score	Precision	Recall (Sensitivity)	F1-Score
0.8246	0.7547	0.6780	0.7143

Discussion

The Extra Trees classifier demonstrated competitive performance with an F1-Score of 0.7143, indicating a good balance between precision and recall. The model achieved a precision of 0.7547, suggesting its ability to minimize false positives, and a recall of 0.6780, showing its capability to capture most of the true positives. The ROC-AUC score of 0.8246 highlights its overall discriminative ability. Compared to other models, Extra Trees benefited from its ensemble nature, providing robustness and reducing variance. Further optimization could explore increasing the number of estimators or fine-tuning the depth to further enhance recall without sacrificing precision.

2.6 Conclusion

Based on the evaluation metrics, Random Forest emerges as the best model to use. It achieves a strong balance between precision (0.737) and recall (0.712), resulting in the highest F1 score (0.724) and a competitive PR AUC (0.796). This suggests that Random Forest is the most reliable in handling both positive and negative classes effectively while maintaining robust performance across all metrics.