Ice Classification in the Greenland

Project Machine Learning, prof. Mathilde Mougeot

Fabio Marcaurelio & Davide Messina

M2QF, ENSIIE & Université Paris-Saclay, 2024/2025

Abstract

The aim of this project is to train multiple machine learning models to address a real-world problem such as melting glaciers. We focused on the three ensemble methods namely Bagging, Boosting and Stacking as well as simpler models. Each of these is then retested taking into account the weights of the classes. The study concludes by highlighting the trade-off between precision and recall in scenarios with class imbalance.

Contents

1 Context

Long-Term Infrasonic Monitoring of Land and Marine-Terminating Glaciers in Greenland

1.1 Target

The objective of this study is to demonstrate how these infrared sensors can pick up signals of displacement and calving of large ice masses over time. In this application we are interested in one potential target (Y1). The displacement of large volumes of air generates low-frequency acoustic waves in the atmosphere. As shown in Figure 1, the blue line represents the shape of the infrared sensor over time. We notice that our variable of interest (Y1) assumes a cyclical shape, with the signal rising during the summer due to an increase in atmospheric temperature, which causes fractures or movements in glaciers. In the Python code, we also plot the cumulative sum to detect periods of higher stress.

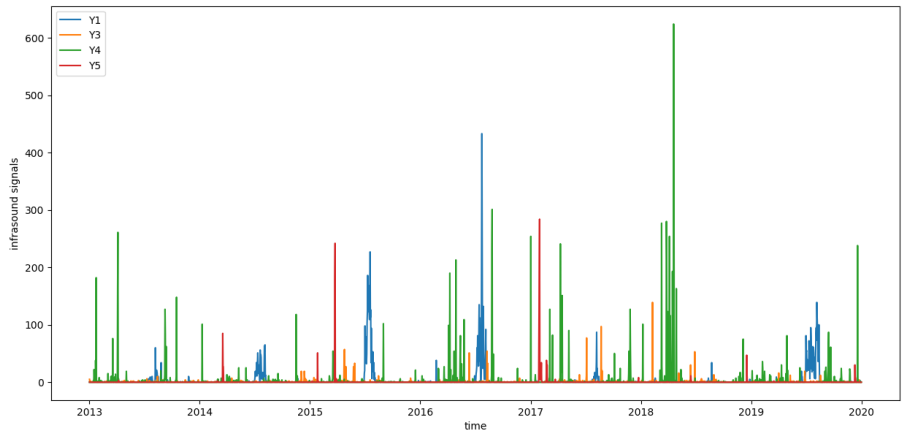

Figure 1

1.2 Features

We have 10 features for training our models:

1. Climate information: The European Weather Center provides information on 2 meter below sea temperature (t2m), Sea-surface temperature (SST), and wind speed (u10, v10);
2. Sea Ice Concentration information (SIC);
3. Greenland liquid water discharge simulated by Region Climate Models for 5 regions (r1_MAR, r2_MAR, r3_MAR, r4_MAR, r5_MAR).

In the Python code, we also plot the shape of these variables. Many of them exhibit seasonal patterns, and in particular, the variables related to water discharges from the glaciers are highly correlated with each other, as seen in the correlation matrix. For each of these variables, we also plotted the distribution to show how even the MAR variables have a zero mean with peaks corresponding to the melting of the glaciers during the summer.

2 Transform the target variable into a binary

As we have seen previously, infrared signals show seasonal movements consistent with the melting of glaciers and therefore the increase in temperatures. Since this is a cyclical movement that occurs only for a few weeks each year, choosing the threshold becomes a crucial aspect of our research. If we were to set the threshold too high, we risk excluding useful information by focusing only on the rarest and most

extreme events, which would shift the classification toward anomaly detection. For this reason, we find it consistent to choose a threshold of zero. From a theoretical perspective, this level allows us to retain all positive infrared signals. Statistically, we have seen in the initial report that the median of Y1 is exactly zero. At this point, we have 294 instances of Y=1 and 2262 instances of Y=0, which represents the maximum number of Y=1 in this dataset. In our opinion, the sample exhibits imbalance issues.

3 Models

We are working in the context of Classification, which is a supervised learning task where the goal is to assign a label to an observation based on its features. To do that, we use predictive classification models. After training the model, its performance is evaluated using metrics such as accuracy, precision, recall, AUC (Area Under the Curve), and the Precision-Recall curve. When evaluating a model's performance, selecting the appropriate metric depends on the context of the problem and the relative costs or benefits associated with specific outcomes. In the case of unbalanced classes, we are more interested in evaluating the so-called Precision-Recall curve rather than the ROC curve, since the former summarizes the trade-off between the true positive rate and the positive predictive value for a predictive model using different probability thresholds.

We want to highlight that for this task, we will focus our research on 6 main models with different characteristics and therefore different possible predictions. After evaluating each model and comparing the results, our attention turns to re-evaluating each of these models, given the possibility that our sample is unbalanced, at which point we can re-evaluate each previous model with the new rebalanced one. We focused on the three types of ensemble learning: Bagging, Boosting, and Stacking. For Bagging, the main idea is to reduce the variance in a dataset, and we applied it to the Decision Tree classifier for this reason. In Boosting, we train a sequence of models; each model is trained on a weighted training set. We assign weights based on the errors of the previous models in the sequence. The main idea behind sequential training is to have each model correct the errors of its predecessor. Stacking enables us to train multiple models to solve similar problems, and based on their combined output, it builds a new model with improved performance.

The goal of our research is <u>NOT</u> primarily to demonstrate whether there is a model that outperforms all others, but to evaluate the models and their behavior in the context of the challenges presented by the sample and understand how to enhance their predictive capacity using the techniques and methodologies we have learned.

3.1 Metrics

The metrics we will take into consideration to evaluate each model are the following:

Confusion Matrix: The confusion matrix is a 2×2 matrix that compares a model's predictions with the actual outcomes, highlighting performance metrics. It helps visualize the trade off between false positives (incorrectly predicted positives, Type I Error) in the top-right corner and false negatives (missed positives, Type II Error) in the bottom-left corner. This tool is crucial for evaluating classification accuracy and balancing prediction errors.

Accuracy: Measures the overall correctness of the model by calculating the proportion of correctly predicted observations:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total Observations}}$$

where TP (True Positives) and TN (True Negatives) represent correctly predicted positive and negative outcomes, respectively.

Precision: Evaluates the quality of positive predictions by identifying how many of the predicted positive outcomes are actually correct:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

where FP (False Positives) are the incorrectly predicted positive cases.

Recall: Assesses the model's ability to correctly identify all actual positive cases. It is also referred to as sensitivity:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

where FN (False Negatives) are the actual positives incorrectly classified as negatives.

F1 Score: Combines precision and recall into a single metric, particularly useful when dealing with imbalanced datasets. It is defined as the harmonic mean of precision and recall:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

AUC-ROC: (Area Under the Receiver Operating Characteristic Curve) is a comprehensive metric that evaluates model performance across all possible classification thresholds, offering insights beyond single-threshold metrics like precision, recall, and F1 Score. The ROC Curve graphically illustrates the trade-off between the true positive rate (TPR) and the false positive rate (FPR) at various thresholds. The AUC metric provides a holistic view of a model's ability to distinguish between classes, making it invaluable for comparing classifiers

Precision-Recall Curve

The **Precision-Recall (PR) curve** is a tool for analyzing a model's performance across various thresholds by plotting recall (x-axis) against precision (y-axis). Derived from the confusion matrix, the curve visualizes the trade-off between minimizing false positives (FPs) and false negatives (FNs). Due to the inverse relationship between precision and recall, the PR curve typically has a negative slope and a non-linear shape.

3.2 Logistic Regression

The model equation is derived by applying the *sigmoid function* to the linear regression output, transforming it into a probability value between 0 and 1:

Logistic Regression Equation:

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n)}}$$

Results:

1. Training and Test score: 0.9452 , 0.9343

2. Confusion Matrix:
$$\begin{bmatrix} 544 & 12 \\ 30 & 53 \end{bmatrix}$$
3. Precision, Recall, F1-Score (Y>0): 0.82, 0.64, 0.72
4. Area Under the Curve (AUC): 0.8925

Discussion:
Logistic regression performs quite well in terms of the trade-off between the training and test sets, indicating no evidence of overfitting, given that the accuracies of the sets are very similar. However, at the same time, the model has unsatisfactory recall due to the high number of false negatives.

3.3 Decision Tree

A decision tree is a flowchart-like structure used for decision-making, where internal nodes represent features, branches define decision rules, and leaf nodes indicate outcomes. The root node, positioned at the top, starts the recursive process of splitting the dataset based on attribute values, a technique called recursive partitioning. This structure visually resembles a flowchart and mimics human decision-making, making it easy to interpret and understand.

1. Training and Test score: 1.0000 , 0.9030
2. Confusion Matrix:
$$\begin{bmatrix} 523 & 33 \\ 29 & 54 \end{bmatrix}$$
3. Precision, Recall, F1-Score (Y>0): 0.63, 0.69, 0.66
4. Area Under the Curve (AUC): 0.7956

Discussion:
We have an issue: the model suffers from overfitting, which is evident from the training score of 1, which is significantly higher and more divergent than the test score of 0.9030. That said, it is unnecessary to discuss the model's results further, so we focused on applying the fine-tuning technique to the maximum depth of the nodes to achieve a more satisfactory test and training score.

3.3.1 Decision Tree (Fine-tuning)

We also want to set a maximum number of nodes that our model can reach. To achieve this, we use the fine-tuning technique to estimate the maximum depth that maximizes the model's accuracy. The Classification and Regression Tree algorithm uses the Gini index to determine split points. The attribute with the smallest weighted impurity, calculated as the sum of impurities across partitions, is chosen for splitting. This structured yet intuitive approach ensures that decision trees are robust and capable of handling various data scenarios effectively.

Results:

1. Training and Test score: 0.9510 , 0.9390
2. Confusion Matrix:
$$\begin{bmatrix} 546 & 10 \\ 29 & 54 \end{bmatrix}$$
3. Precision, Recall, F1-Score (Y>0): 0.84, 0.65, 0.73
4. Area Under the Curve (AUC): 0.8779

Discussion:
Now the model performs better, and the overfitting problem is solved. We can observe that all the values in the classification report improve compared to those of logistic regression.

3.3.2 Decision Tree (Bagging)

Bagging (Bootstrap Aggregating) is an ensemble learning method designed to improve model robustness and accuracy by reducing variance and mitigating overfitting. It operates through the following key steps:

- **Bootstrap Sampling:** Multiple models are independently trained on random subsets of the data, sampled *with replacement*. These subsets, called bootstrap samples, allow individual data points to appear in multiple samples, ensuring diversity among the models.
- **Training:** Each model learns patterns from its unique bootstrap sample, capturing different aspects of the dataset.
- **Aggregation:** Predictions from all models are combined, typically via averaging (for regression) or majority voting (for classification). This aggregation smooths out individual model errors, leveraging their strengths.

Bagging is particularly effective when the base models exhibit high variance, as it stabilizes predictions and reduces overfitting. It ensures that the final ensemble model is both accurate and robust by incorporating diverse perspectives from its constituent models.

Results:

1. Training and Test score: 0.9515 , 0.9390
2. Confusion Matrix:
$$\begin{bmatrix} 545 & 13 \\ 28 & 55 \end{bmatrix}$$
3. Precision, Recall, F1-Score (Y>0): 0.83, 0.66, 0.74
4. Area Under the Curve (AUC): 0.8858

Discussion:

Applying bagging to our previous model does not introduce any major changes to the results. In both models, r4_MAR is detected as almost the only important variable, despite the large variance among these variables.

3.4 Random Forest

Random Forest is a supervised machine learning algorithm designed for both classification and regression tasks, though here we focus on its application in classification. It is an ensemble method, combining the predictions of multiple decision trees to improve performance and reduce overfitting. In a Random Forest, several decision trees are trained on different random subsets of the data and features. Each tree independently provides a prediction (or "opinion") for a given input. The final classification is determined by aggregating these predictions, typically using a majority vote to select the most common outcome. This method enhances model robustness and accuracy by leveraging the diversity among decision trees.

Results:

1. Training and Test score: 0.9489 , 0.9358
2. Confusion Matrix:
$$\begin{bmatrix} 544 & 12 \\ 29 & 54 \end{bmatrix}$$
3. Precision, Recall, F1-Score (Y>0): 0.82, 0.65, 0.72
4. Area Under the Curve (AUC): 0.8876

Discussion:

Since the random forest is a model that works in parallel, like bagging, its goal is to reduce the model's

variance, and as we can see, it does this very well. The results are very similar to those of the previous models. Compared to the decision tree, this model distributes the importance among all the water discharge variables. This difference may be due to the way the two models handle the problem of covariance between variables. By fine-tuning the model, we noticed that setting the maximum depth to 3 turned out to be the best choice for the model.

3.5 The Naive Bayes Classifier

In the Naive Bayes Classifier, we define:

- Y: the target variable (the class label we want to predict).
- X: the explanatory multivariate variable (the set of features used to make the prediction).

The core of the Naive Bayes method is the application of Bayes' Theorem for calculating the posterior probability $P(Y = y|X = x)$, which expresses the probability of a class y given a set of input features x. Bayes' formula is given by:

$$P(Y = y|X = x) = \frac{P(X = x|Y = y) \cdot P(Y = y)}{P(X = x)}$$

Results:

1. Training and Test score: 0.9218 , 0.9343
2. Confusion Matrix:
$$\begin{bmatrix} 532 & 24 \\ 18 & 65 \end{bmatrix}$$
3. Precision, Recall, F1-Score (Y>0): 0.73, 0.78, 0.76
4. Area Under the Curve (AUC): 0.8534

Discussion:

As we can see, this model, without overfitting issues, gives us a good level of predicted true positives and, therefore, a high recall at the expense of lower precision.

3.6 Handle Imbalanced dataset

Class weighting is a technique that emphasizes the minority class during model training by assigning it higher weights. This adjustment influences the **loss function**, rather than the individual predictions, guiding the model to prioritize underrepresented classes during optimization. While effective in addressing class imbalance, care must be taken to avoid overcompensation, which could lead to the model neglecting other classes entirely and, thereby, skewing predictions. Proper balance is key to ensuring fair representation without compromising overall performance.

3.6.1 Logistic Regression ACW

1. Training and Test score: 0.9358 , 0.9437
2. Confusion Matrix:
$$\begin{bmatrix} 539 & 17 \\ 19 & 64 \end{bmatrix}$$
3. Precision, Recall, F1-Score (Y>0): 0.82, 0.64, 0.72
4. Area Under the Curve (AUC): 0.9013

3.6.2 Decision Tree (Fine-tuning) ACW

1. Training and Test score: 0.9264 , 0.9218

2. Confusion Matrix:
$$\begin{bmatrix} 525 & 31 \\ 19 & 64 \end{bmatrix}$$
3. Precision, Recall, F1-Score (Y>0): 0.82, 0.64, 0.72
4. Area Under the Curve (AUC): 0.8449

3.6.3 Random Forest ACW

1. Training and Test score: 0.9353 , 0.9390
2. Confusion Matrix:
$$\begin{bmatrix} 546 & 19 \\ 20 & 63 \end{bmatrix}$$
3. Precision, Recall, F1-Score (Y>0): 0.77, 0.76, 0.76
4. Area Under the Curve (AUC): 0.8890

Discussion Models ACW:
We note from the results of the three models above that, for all of them, the gap between the training and test sets has improved significantly, and therefore, the variance has been reduced. We note from the results of the three models above that, for all of them, the gap between the training and test sets has improved significantly, and therefore, the variance has been reduced.

3.6.4 AdaBoost

We added AdaBoost to implement a sequential model in our study. The AdaBoost algorithm is a boosting technique used as an ensemble method in machine learning. It is called Adaptive Boosting because the weights are re-assigned to each instance, with higher weights assigned to incorrectly classified instances. We understand that this method behaves differently from Bagging. The results of this model are very similar to the previous ones in terms of predictions with a very low gap between the training and test scores..

3.6.5 Stacking

Stacking is a machine learning strategy that combines the predictions of numerous base models, also known as first-level models or base learners, to generate a final prediction. It involves training numerous base models on the same training dataset, then feeding their predictions into a higher-level model, also known as a meta-model or second-level model, to generate the final prediction. The main idea behind stacking is to combine the predictions of different base models in order to achieve superior predictive performance compared to using a single model.

4 Conclusion

K-Fold Cross-Validation
Giving the K-Fold Cross-Validation we can compare different models for difference scores. K-Fold Cross-Validation is a reliable method for assessing machine learning model performance. It divides the dataset into k subsets (folds), using one fold for testing and the remaining $k-1$ folds for training in each iteration. This process repeats k times, ensuring every fold serves as a test set once. By averaging the results, it provides a comprehensive evaluation, reducing the risk of overfitting and improving generalization to unseen data.
Since we are interested in precision and recall, as previously stated, we will not show here the summary

of accuracy, the previous data tell us that all models that do not have overfitting problems have a very high level of accuracy. We are more interested in demonstrating how each model behaves with respect to the precision-recall trade off.

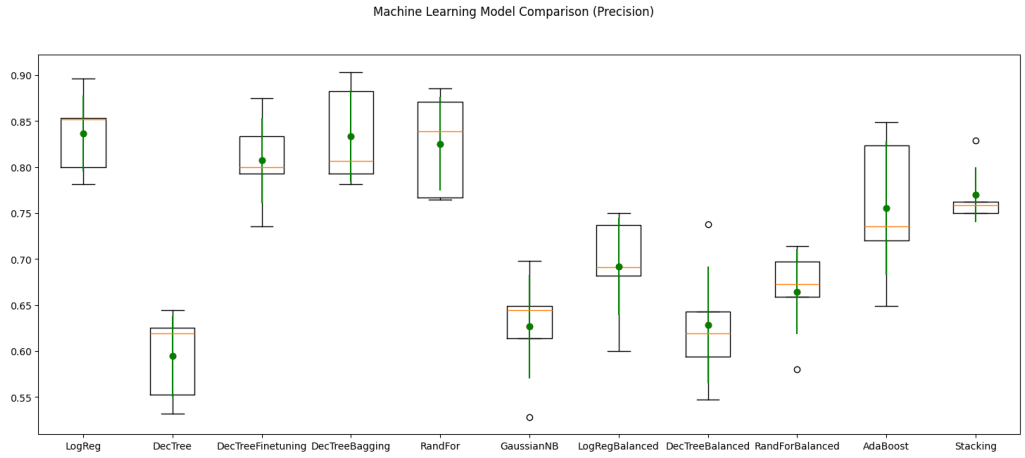

Figure 2

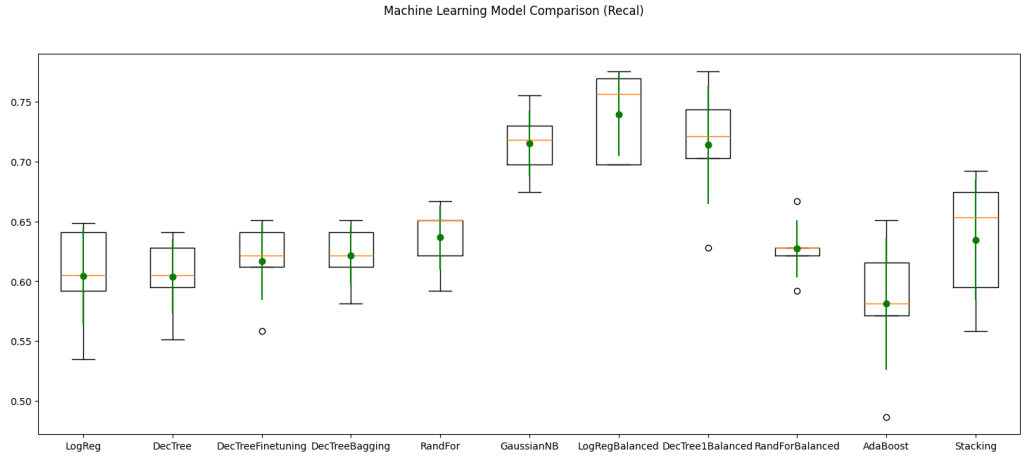

Figure 3

As we can see from the graphs above, the models that have not been adjusted with class weights (the first 6 from the left) have high precision compared to lower recall. Conversely, the three subsequent adjusted models exhibit the opposite characteristics just as we wanted to highlight. For the two final models, AdaBoost and Stacking, these show more weighted results than the others. As mentioned earlier, the accuracy data is somewhat misleading due to the class imbalance problem, as class zero has a much higher weight. In fact, the accuracy of each model is very high. For this reason, even though we have highlighted the data in the code, we are not including it in our report. For the same reason, the AUC is not very explanatory in our case, unlike the precision-recall curve. We can conclude by saying that there are essentially two types of models: those that take into account class imbalance and those that do not. The ability to predict true positives is greater for balanced models than for basic models, at the expense of a lower level of precision due to the increase in false positives.