Final report

Course of Machine Learning

Authors: *Antonio Antona & Naïm Souni*

Teacher:	*Mathilde Mougeot*
Department:	Applied mathematics
Date:	Novermber 7,2022

Contents

1 Introduction **1**
- 1.1 Description of the Output Variable . . . 1
- 1.2 Description of the Input Variables . . . 1
- 1.3 Data . . . 1

2 Analysis **2**
- 2.1 Analysis of the target column. . . . 2
- 2.2 Analysis of our dataset . . . 3
  - 2.2.1 Correlation between our variables defined by a fixed threshold . . . 3

3 Modeling **4**
- 3.1 Three different models . . . 4
- 3.2 Evaluation . . . 5
  - 3.2.1 Learning curve: . . . 5
  - 3.2.2 F1-score: . . . 5
  - 3.2.3 AUC: . . . 6
- 3.3 Interpretation . . . 6

4 How to improve results **8**
- 4.1 Voting classifier . . . 8
- 4.2 Stacking . . . 8

5 Conclusion **8**

1 Introduction

The aim of this study is to evaluate the ability of several learning machines for building a predictive classifier to evaluate, thanks to infrasound records, the (low or high) quantity of ice at a given spot in the Groenland. In this application, there are 4 potential target ouputs corresponding to infrasound signals (Y1, Y3, Y4, Y5) (file Targets.csv). The covariables are defined by 9 variables providing by the European Centre for Medium-Range Weather Forecasts (ECMWF) .

1.1 Description of the Output Variable

The displacement of large volumes of air leads to low-frequency acoustic waves in the atmosphere. This so-called infrasound is inaudible to humans as it has frequencies lower than 20 Hz. In the raw data, the 4 output variables are quantitative infra-sound records. For this classification study, the quantitative valued target signals will be transformed into binary information using a approriate given threshold.

1.2 Description of the Input Variables

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

More detailed information can be found in the Geophysical Research Letters, "Long-Term Infrasonic Monitoring of Land and Marine- Terminating Glaciers in Greenland", Research letter, DOI 10.1029/2021GL097113, AGU advancing earth and space conference.

1.3 Data

The following instructions let to read the Input (data_features.csv) and output (data_Targets) data in Python.
Description of the work

- In this work, we studied only one given target variable (*Y4:* G3, G7, G11, G15, G19, G23, G27).
- We transformed the target variable into a binary variable using an appropriate and motivated threshold.
- We did a train, test and comparing of different binary machine learning classifiers to predict the binary target variable.
- We made a conclusion about the performances and the models used.

2 Analysis

2.1 Analysis of the target column.

Transform the target variable into a binary variable using an appropriate and motivated threshold. We first want to know how the column target is distributed,whether the distribution is made uniformly or disproportionately. For doing so we plot a pie chart histogram figure 1.

```
0      2054
1       192
2        58
3        33
4        27
        ...
49        1
82        1
241       1
150       1
238       1
Name: Y4, Length: 78, dtype: int64
<matplotlib.axes._subplots.AxesSubplot at 0x7f6e6f478310>
```

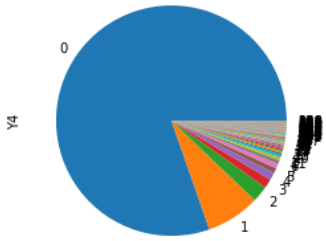

Figure 1

So we can see that the data is very poorly distributed. This is because most of the data in the target variable is 0 (about 75%) . That's why in order to have a binary column and more uniformly distributed, we decided to set a threshold to 1 and then create a column distributed with only 0 and 1 as we can see in the histogram

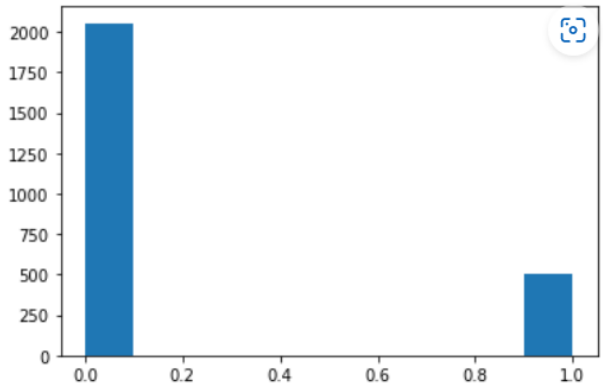

Figure 2

2.2 Analysis of our dataset

First, we decided to delete the variable with time because it creates some problem with the models that we will use after. Then, we divide our dataset into a train and a test set in order to train our model and evaluate it with some data that the model has never seen before. We decided to divide a train set into 80% of our data (2044 rows)and 20% for the test set (512 rows). Finally with the Standard Scaler transformer, we will scale our data so that our models will not be biased. Notice that with the random Forest model we don't have to use Scandard Scaler because this model has no idea of order of magnitude.

2.2.1 Correlation between our variables defined by a fixed threshold

Figure 3: Analysis of the correlation between our variables defined by a fixed threshold

Thus, we observe that the variables 'r2_MAR', 'r3_MAR', 'r4_MAR', 'r5_MAR' seem particularly correlated and have even exceeded our maximum correlation threshold fixed at 0.9. Thus several options are available to us: The first is to take this correlation into account and build a new variable New Variable containing the average of the correlated variables and then delete the variables mentioned. However, doing this would deprive us of a lot of data that can be useful in our model. This is why we decide not to immediately remove these variables.

3 Modeling

3.1 Three different models

1. *Random Forest:* like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. The fundamental concept behind random forest is that a large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

2. *SGD Classifier:* is a Linear classifier with SGD (stochastic gradient descend) training. Which linear classifier is used is determined with the hypter parameter loss. So, if I write clf = SGDClassifier(loss='hinge') it is an implementation of Linear SVM and if I write clf = SGDClassifier(loss='log') it is an implementation of Logisitic regression.

3. *SVM Classifier:* The basics of Support Vector Machines and how it works are best understood with a simple example. Let's imagine we have two tags: red and blue, and our data has two features: x and y. We want a classifier that, given a pair of (x,y) coordinates, outputs if it's either red or blue. We plot our already labeled training data on a plane.

A support vector machine takes these data points and outputs the hyperplane (which in two dimensions it's simply a line) that best separates the tags. This line is the decision boundary: anything that falls to one side of it we will classify as blue, and anything that falls to the other as red.

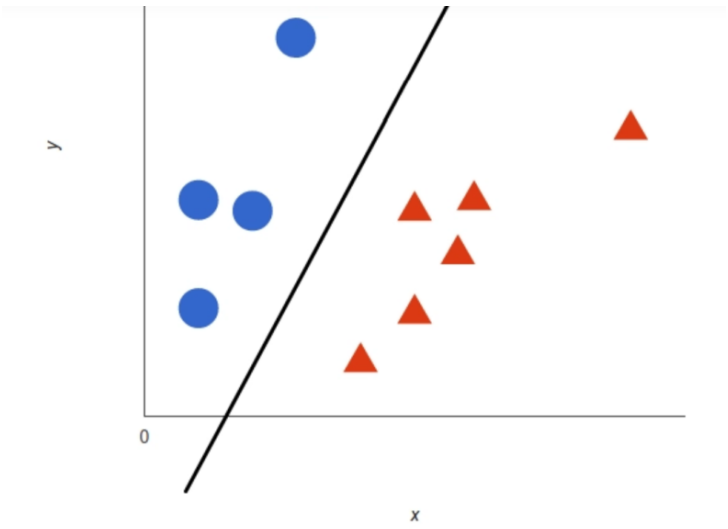

Figure 4

3.2 Evaluation

For evaluating our models, we will use the f1 score. And in order to have an idea of how our results improved over time, we use learning curves.

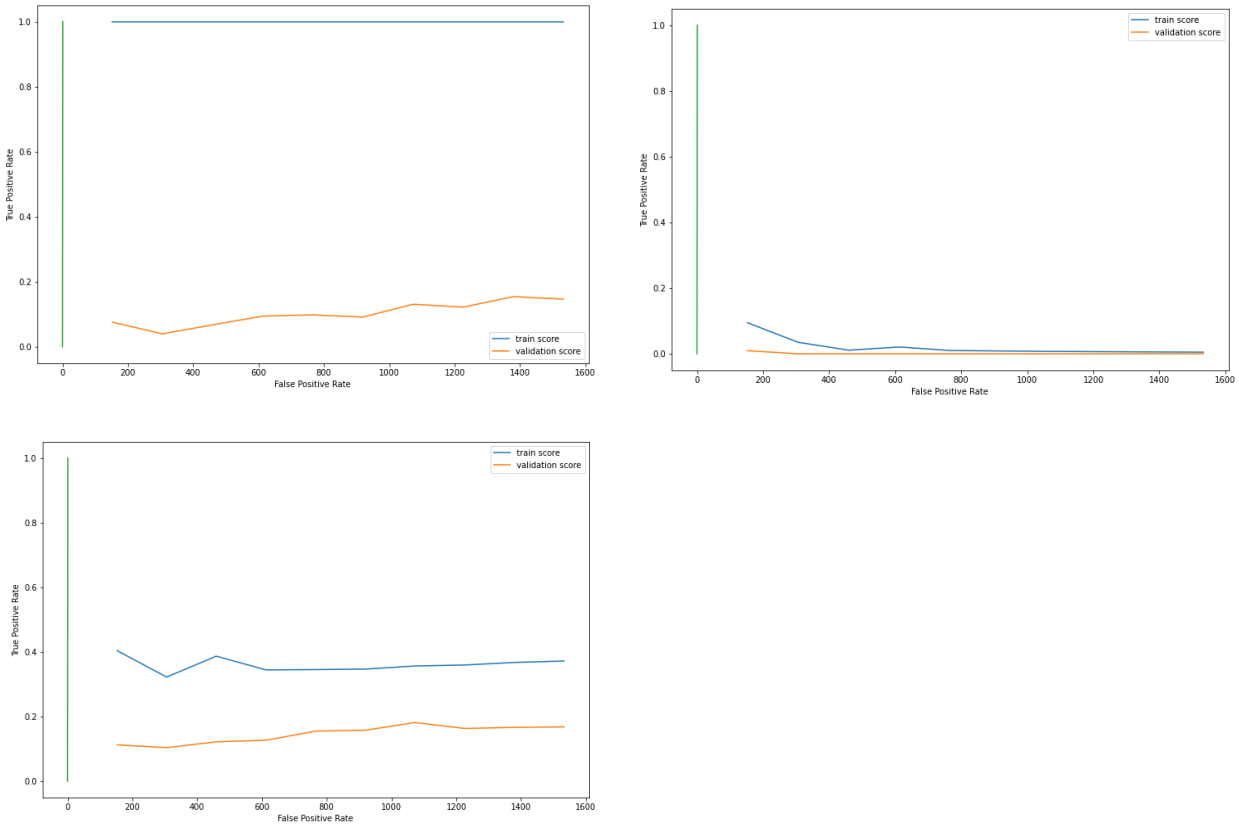

Figure 5

3.2.1 Learning curve:

A learning curve is a plot of model learning performance over experience or time. Learning curves are plots that show changes in learning performance over time in terms of experience. Learning learn from a training dataset incrementally. The model can be evaluated on the training dataset and on a hold out validation dataset after each update during training and plots of the measured performance can created to show learning curves

3.2.2 F1-score:

The f1-score gives us the harmonic mean of precision and recall. The scores corresponding to every class will tell us the accuracy of the classifier in classifying the data points in that particular class compared to all other classes. The support is the number of samples of the true response that lie in that class.

3.2.3 AUC:

The target variable is imbalance so we proceed with the AUC.
The ROC AUC is sensitive to class imbalance in the sense that when there is a minority class, you typically define this as the positive class and it will have a strong impact on the AUC value. This is very much desirable behaviour.
Accuracy is for example not sensitive in that way. AUC stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1). AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example.

3.3 Interpretation

We can see that Random Forest is a model who is overfitting. This can be seen in particular because the score of the train score is much higher than that of the validation score.
So if we can into account the f1 score to evaluate our models, we conclude that Random Forest seems to be the best for predicting both 0 and 1 results. Indeed, since we have a lot more 0 than 1 as we said in the first part, we can conclude notice logically that our model obtain a good score on predicting 0 and a less good result for predicting 1. Random forest seems to be the best out of the model used for predicting both 0 and 1 (regarding the f1 score)

RandomForest
[[402 9]
[86 15]]

	precision	recall	f1-score	support
0	0.82	0.98	0.89	411
1	0.62	0.15	0.24	101
accuracy			0.81	512
macro avg	0.72	0.56	0.57	512
weighted avg	0.78	0.81	0.77	512

KNN
[[387 24]
[89 12]]

	precision	recall	f1-score	support
0	0.81	0.94	0.87	411
1	0.33	0.12	0.18	101
accuracy			0.78	512
macro avg	0.57	0.53	0.52	512
weighted avg	0.72	0.78	0.74	512

SVM
[[411 0]
[101 0]]

	precision	recall	f1-score	support
0	0.80	1.00	0.89	411
1	0.00	0.00	0.00	101
accuracy			0.80	512
macro avg	0.40	0.50	0.45	512
weighted avg	0.64	0.80	0.71	512

Figure 6: Models

The precision and the recall of the SVM is equal to 0 (it predicts only class 0)

Find out the best parameters of our Random Forest
Random Forest use different hyperparameters to predict the results.

Here are the parameters used by default by this model :
'bootstrap': True,
'ccp_alpha': 0.0,
'criterion': 'mse',
'max_depth': None,
'max_features': 'auto',
'max_leaf_nodes': None,
'max_samples': None,
'min_impurity_decrease': 0.0,
'min_impurity_split': None,
'min_samples_leaf': 1,
'min_samples_split': 2,
'min_weight_fraction_leaf': 0.0,
'n_estimators': 100,
'n_jobs': None,
'oob_score': False,
'random_state': 42,
'verbose': 0,
'warm_start': False

However we can try to optimise these parameters using the function gridSearch.
The most important arguments in RandomizedSearchCV are n_iter, which controls the number of different combinations to try, and cv which is the number of folds to use for cross validation (we use 100 and 3 respectively). More iterations will cover a wider search space and more cv folds reduces the chances of overfitting, but raising each will increase the run time.
Machine learning is a field of trade-off and performance vs time is one of the most fundamental. Even if the code require a long time to compile, we finally find out what are the best hyperparameter to use for having a better result using this model.
Here are the best hyperparameters that we find with this method :

'n_estimators': 400,
'min_samples_split': 2,
'min_samples_leaf': 4,
'max_features': 'sqrt',
'max_depth': 10,
'bootstrap': True

4 How to improve results

4.1 Voting classifier

From the SGD Classifier, Decision Tree Classifier and KNeighbors Classifier models, we count each prediction of the model as a vote and we finally choose the one that has obtained the most votes.
So we can see that we get a slight improvement with the Voting Classifier compared to our previous models.

```
SGDClassifier 0.765625
DecisionTreeClassifier 0.71875
KNeighborsClassifier 0.775390625
VotingClassifier 0.78515625
```

Figure 7

4.2 Stacking

We train a machine learning model on top of the crowd predictions (the SGD, Tree, KNN models). From the different models used, we train a model (here KNeighbors Classifier) to predict the errors of our previous models to get the final result.

```
model = StackingClassifier([('SGD', model),
                            ('Tree', model),
                            ('KNN', model)],
                             final_estimator=KNeighborsClassifier())

model.fit(X_train, y_train)
model.score(X_test, y_test)
```

```
0.771484375
```

Figure 8

The results are not much better than for the KNeighbors Classifier

5 Conclusion

We can conclude that by using different models we can finally build a performant model and have good results. However, using a threshold to 1 can also not be very convenient in the real life for doing some more precise prediction. That's why we may now should use another model which can be train to predict the values distributed between 1 and 238 to be more convenient in the "real life".