Machine Learning
homework 1

Saverio Taliani 1841465

December 9, 2022

Contents

1 Introduction

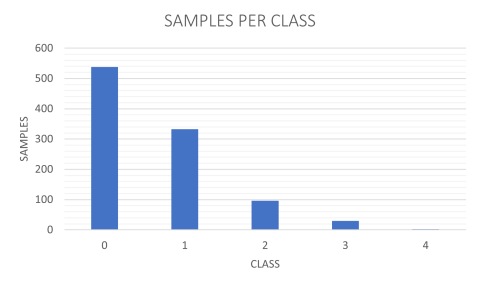
The following discussion has been developed as a solution for the proposed task. The latter consisted in creating a machine learning algorithm able to classify the number of collisions, given a dataset representing information of five drones, such as position x and y, speed, and target position.
In the second part a regression algorithm is presented to solve the problem of finding the Closest Point of Approach (CPA) for the given drones.

2 Classification

2.1 Data Processing

2.1.1 Normalization

The dataset was presented as a .tvs file containing a 1000×37 matrix. For each of the five drones there was seven columns describing the yaw angle, the position, the velocity and the target position in on a plane.
To retrieve information, it was applied the Pandas framework, as a result the dataset was successfully imported to Colab. The first process was aimed to expose the nature of the data, the classes where five representing the possible collision of the drones, the most interesting aspect was that the number of samples for each class was heavily unbalanced.



Then the data were normalized, two approaches are proposed.
By using the pre-processing tools from the scikit library, by imposing "axis=0" the l2 norm is applied for each feature.

$$x = \begin{bmatrix} x_1 \\ ... \\ x_n \end{bmatrix} \rightarrow \|x\| = \sqrt{\sum_{i=1}^{n} x_i^2} \tag{1}$$

Another "Home-made" normalization was implemented by bring the problem in a five dimensional cube with each axes of length one.

$$x_i = \frac{x_i}{\max x_{feature}} \tag{2}$$

Where x_i is a generic entry of the dataset and $x_{feature}$ is the maximum value of the correspondent physical dimension ex. v_x, v_y, angle, ...

However as will be noticed in the following discussion this approach does not perform as well as the previous one, although it brought some improvement with the respect to a raw data analysis.

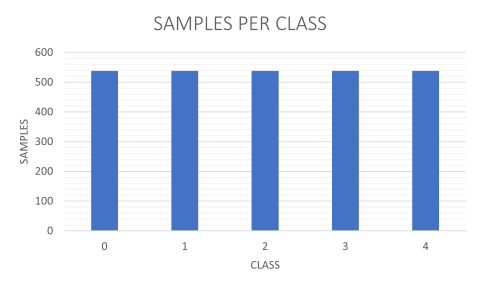
2.1.2 Augmentation

Since the dataset was extremely unbalanced, no model was able to reach an acceptable accuracy with the data as given. However, an interesting operation to cope with such unbalanced sets is the so-called oversampling.

There exist various techniques to execute the oversampling, but the simplest one is the "random oversampling".
Random oversampling is a technique that, given a bunch of data, will randomly pick a subset of it and copy it in such a way to make the count of each class in the dataset equal to the others. This simple approach could still be very effective if the considered samples are sufficiently descriptive for the problem. Unfortunately, this simple method is not enough in this case since the class with less samples contains only three entries which are clearly too few to describe it properly, the method brings the fitting process to a clear overfitting.
An extremely more powerful method for oversampling is the Synthetic Minority Oversampling Technique (SMOTE).
SMOTE generates new samples by linear interpolation, it sets the minority class set A, for each $x \in A$ the k-nearest neighbors of x are obtained by calculating the Euclidean distance between x and every other sample of A. The sampling rate N is set according to the imbalanced proportion, for each $x \in A$, N samples are randomly selected from its k-nearest neighbors, the construct the set A_1. For each example $x_k \in A_1, k = 1, .., N$, the new sample is generated as:

$$x' = x + rand(0, 1) \cdot |x - x_k| \tag{3}$$

in which $rand(0, 1)$ is a random number between 0 and 1.
After the execution of this method the sample count becomes:



Clearly the resampled dataset is now balanced but not only, as it is shown in the following discussion it gives also an outstanding increment of the classification performance.

2.2 Model choice and fit

To perform the classification several tests were performed, in the following analysis will be discussed the nature of the most successful model.
The first attempt was to employee the support vector machine (SVM), given a set of points this method tries to maximize the margin from samples. Considering a linearly separable dataset, and considering h as the hyperplane which separates the samples, the margin is:

$$\frac{|y(x_k)|}{\|w\|} \tag{4}$$

Where x_k is the closest point of the dataset to h. In addition, each sample had the form of a long array containing information of separated bodies, to deal with this problem has been necessary to use the kernelized version of SVM.
Kernels are real-valued functions $k(x, x')$ symmetric and non-negative, which are meant to be similarity measures between samples. At this moment a Grid search operation was operated to choose the best fitting parameters for the model. A detailed description of the outcomes is provided in the evaluation section.

A notable result in the classification task come from the implementation of a Random Forest.
Random forest is an ensemble method which generates a set of decision trees and estimate the class of a sample by collecting votes from each generated tree. A Decision Tree is a simple classifier, in practice a tree structure is built to test the attributes of a sample a node corresponds to the attribute and the

branches to its possible values. Evaluating a sample becomes a task of following a path on the tree, when a leaf is reached it indicates its class.
Again, to find the best parameters for the Random Forest algorithm such that, number of estimators or the max depth of the trees, a grid search has been employed and its result will be described in the next session.

2.3 Evaluation

The evaluation of the classification performance has been performed through the employee of several instruments. Talking about learning the first metrics that comes to mind is the accuracy. By applying the normalization per feature from sci kit library and without adopting any oversampling strategy the executed tests brought the following results:

1. SVC ,kernel="rbf", C=10, average accuracy: 0.456
2. Naive Bayes, priors=None, var smoothing=10^{-9} : 0.45
3. Adaboost with decision tree, n estimators=100: 0.43
4. LinearSVC penalty=l2, loss=squared hinge: 0.489
5. Random Forest, max depht=5 , n estimators=200: 0.539

These accuracies are not really satisfying and that's way there was the necessity to work a bit in the pre-processing part to try to make data more suitable to be fit. Anyway, as already mentioned the dataset is unbalanced and for this reason the accuracy is not a proper metric to analyze. There exist others more interesting way to deal with unbalanced data, in the next analysis accuracy will only be cited but the important indexes will be the balanced accuracy, precision, recall and f1-score. Balanced accuracy is calculated as:

$$BA = \frac{1}{2}(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}) \tag{5}$$

Precision is the ability to avoid false positive, in the multi-class case give two sets A, B:

$$P(A,B) = \frac{|A \cap B|}{|B|} \tag{6}$$

On the contrary the recall is the ability to avoid false negatives, in multi-class classification given two sets A, B:

$$R(A,B) = \frac{|A \cap B|}{|A|} \tag{7}$$

The f-1 score is a weighted harmonic mean if the precision and recall, it is calculated as:

$$F_1 = 2\frac{P \cdot R}{P+R} \tag{8}$$

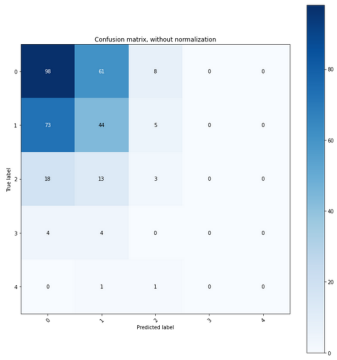
By working on the dataset without any augmentation action, the results are again disappointing. The introduced metrics revealed that the results mentioned before in term of accuracy where quite optimistic, the reality is that the models predict practically only class 0 that is by far the most common. In these conditions the models are not able to classify properly the dataset, a feeling of this was given by the low accuracy, but even lowest values of balanced accuracy, precision and recall for each class and the plot of the confusion matrix gives an exact picture of how far the models are from understanding the situation. Here are presented the values obtained with the same models and parameters as before but run with a split of 0.3 test set, and 0.7 train set.

SVC:

Accuracy 0.435
Balanced accuracy 0.207

	precision	recall	f1-score	support
0	0.508	0.587	0.544	167
1	0.358	0.361	0.359	122
2	0.176	0.088	0.118	34
3	0.000	0.000	0.000	8
4	0.000	0.000	0.000	2
accuracy			0.435	333
macro avg	0.208	0.207	0.204	333
weighted avg	0.404	0.435	0.417	333

(a) Metrics



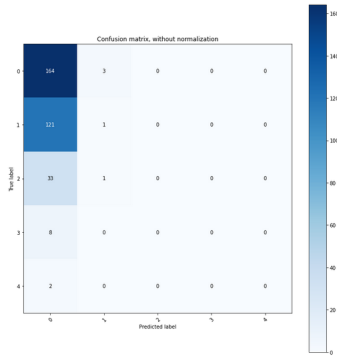
(b) Confusion metrics

As we can see the real situation is that the balanced accuracy is very low 0.287 and precision and recall and f1-score for classes $3, 4$ are just null because they are not even predicted, the class best class is 0 but is not satisfactory anyway because as is evident in the confusion matrix it is not able to distinguish from 0 collision and 1 collision.
By employing other models, the situation does not get better, Naive Bayesian is slightly better in distinguishing class 0 from the others improving a bit its recall, but behave worse than SVC with classes $1, 2$.
Linear SVC behaves pretty much as a binary classifier predicting practically only classes $0, 1$ and classifying quite everything as class 0.
Adaboost has by far the strangest behavior because wrongly predicts some samples from classes $0, 1$ as class 3 and does not predict any sample as class 2. It is not evident how this could happen, but a guess could be that the generated decision trees are not able to distinguish well attributes of that class that maybe are quite like the ones from class 1 and 3.
At the end not surprisingly, there is a poor performance by Random Forest:

Accuracy 0.495
Balanced accuracy 0.198

	precision	recall	f1-score	support
0	0.500	0.982	0.663	167
1	0.200	0.008	0.016	122
2	0.000	0.000	0.000	34
3	0.000	0.000	0.000	8
4	0.000	0.000	0.000	2
accuracy			0.495	333
macro avg	0.140	0.198	0.136	333
weighted avg	0.324	0.495	0.338	333

(c) Metrics



(d) Confusion metrics

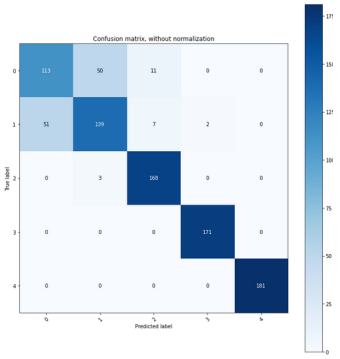
The accuracy given by Random Forest is the highest but as explained before this is not indicative in unbalanced problems, in fact the accuracy that seems slightly better actually is only predicting the most common class.
Now results using oversampling technique are presented, as explained in the relative section they perform strongly better.

With the same parameters, SVC had the outcome:

Accuracy 0.862
Balanced accuracy 0.866

	precision	recall	f1-score	support
0	0.689	0.649	0.669	174
1	0.724	0.698	0.711	199
2	0.903	0.982	0.941	171
3	0.988	1.000	0.994	171
4	1.000	1.000	1.000	181
accuracy			0.862	896
macro avg	0.861	0.866	0.863	896
weighted avg	0.858	0.862	0.859	896

(e) Metrics



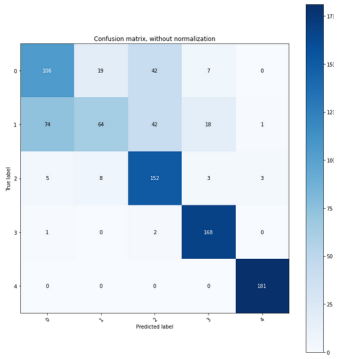
(f) Confusion metrics

This result seems to be outstanding, but a closer look to their results appears a little suspicious, the classes that now are predicted better with the higher f-1 score are classes $2, 3, 4$ while classes $0, 1$ are often miss predicted. This fact conducts to the hypothesis that the model overfits with the new generated samples. In fact, performing the oversampling only to the test set, brings back to poor performance.
Another approach which should reduce overfitting is by using Random Forest which should prevent it especially if instructed with a pruning so a max depth.

Accuracy 0.749
Balanced accuracy 0.760

	precision	recall	f1-score	support
0	0.570	0.609	0.589	174
1	0.703	0.322	0.441	199
2	0.639	0.889	0.743	171
3	0.857	0.982	0.916	171
4	0.978	1.000	0.989	181
accuracy			0.749	896
macro avg	0.749	0.760	0.736	896
weighted avg	0.750	0.749	0.729	896

(g) Metrics



(h) Confusion metrics

Unfortunately, the problem of overfitting is not solved, classes which underwent a higher augmentation are predicted better, the others are not kept very well.

The same analysis has been repeated by employing the 'homemade' normalization and without any normalization. The outcomes of these are like what were presented above, there are no comments to add. The main difference is only in the overall accuracy that is always lower especially for raw data.

3 Regression

If the classification task was not straightforward the regression one was even trickier.
The goal was to find the minimum Closest Point of Approach (CPA) for each sample. The CPA is the occurrence of minimum range between two drones of which at list one is moving.

3.1 Data processing

The dataset was the same as before and the analysis is similar.

3.1.1 Normalization

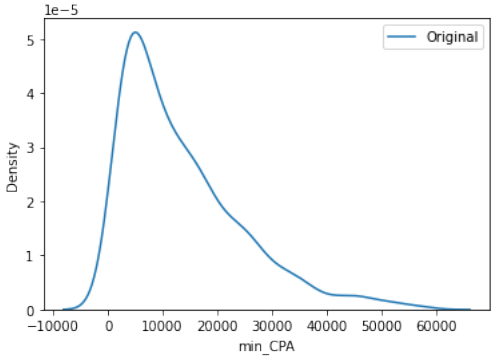
The main difference with the classification is that when the normalization is applied it is applied also to the output column.
To test the regression, it was applied the same l2 norm mentioned in the classification.
Another normalization considered was intended to have all values x such that $x \in [0, 1]$:

$$x = \frac{x - min(x)}{max(x)} \tag{9}$$

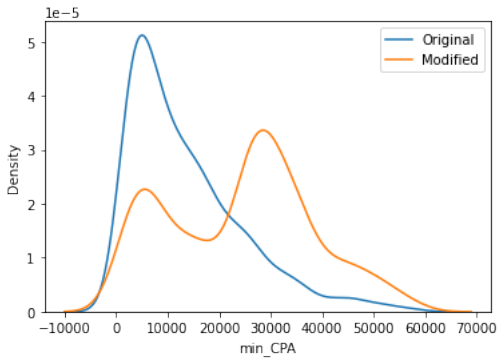
This procedure is repeated for each column of the dataset.

3.1.2 Smoter

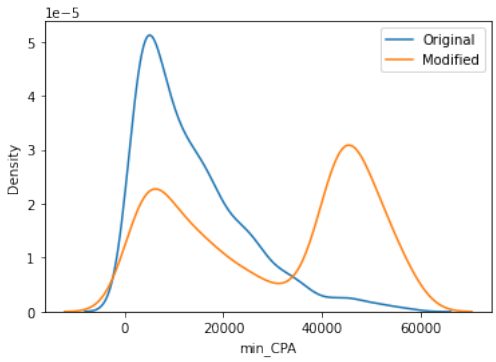
Like in the first task the problem was unbalanced, a quick analysis of this task proved that the distribution of the outcome was uneven among all possible values taken.



It is evident that higher values of CPA are extremely rare with the respect to smaller ones. By using the library smogn it is possible to replicate what SMOTE does for a classification task.
The application of this algorithm changes the distribution of samples in the dataset.



(i) Metrics



(j) Confusion metrics

Two different setting were tested. In image (i) the generation of new sample was balanced, at the end samples were even less than before (850), but as it is visible the distribution as more than one peak.
In image (j) is described the distribution given by a much more aggressive generation that brought the dataset to become composed by 1931 samples.

3.2 Models

Taking in consideration that no model was able to reconstruct really the relation between inputs and output, the most notable results were given by K-nearest neighbors, linear regression and Adaboost regressor.

K-neighbors the learning of this algorithm is based on the k nearest neighbors of each query point, to each of these points is associated a weight. The easiest way is by putting uniform weights, otherwise the method can consider the distance of each neighbor from the new considered sample.

The linear regressor fits a linear model with coefficients $w = (w_1, ..., w_2)$ to minimize the residual sum of squares between observed targets in the dataset and those predicted by the linear approximation.

$$\min_w \|Xw - y\|_2^2 \tag{10}$$

Adaboots instead is an ensemble method, given a regression model it will fit a sequence of weak learners on different data. The predictions from all of them are then combined through a weighted majority vote. The boost in this procedure is the addition of weights to each of the training samples that are initially initialized as $w_i = \frac{1}{N}$ and then changes at each iteration according to correctness of the predictions. In this case as a base regressor it was employed a Decision tree.

Among all other tests it has been tried also the SVR algorithm with various kernels, it's worth to mention the tentative of employ of a personalized kernel. The idea was to compare two entries of the dataset with the variance of the data. Since the largest values were usually positions a large variance in the sample should in same way in some way represent how distant the drones were. Unfortunately, the application of this approach as it will be shown in the next section was not very successful.

3.3 Evaluation

The evaluation of the regression performance comes through the analysis of two important indices the r-2 score and an error function.

The r-2 score represents the proportion of variance of the output that has been explained by the independent variables in the model. It measures the goodness of new samples through the proportion of variance.

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})^2}{\sum_{i=1}^{n}(y_i - \bar{y_i})^2} \tag{11}$$

where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$
This value should be in $(-\infty, 1]$ range but usually is scaled to $[0, 1]$, so negative values are actually valid, and they just mean that the approximation of the target function is not good.

Another way to visualize the actual capability to predict the actual output of a sample given a new entry is by seeing the error between the prediction and the actual output. Usually, it is used the mean square error, but in this case, it would not have any physical meaning, so to describe the prediction error two metrics have been employed:

- Root mean square error: $RMSE(y, \hat{y}) = \frac{1}{n_{samples}}\sqrt{\sum_{i=0}^{n_{samples}-1}(y_i - \hat{y_i})^2}$

- Mean absolute error: $MAE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} |y_i - \hat{y}_i|$

Both these methods applied to this peculiar problem have an outcome in meters that can be immediately interpreted.
Going to the results, by applying only the normalization the results are incredibly bad with each possible regressor.
The best results are obtained with k-neighbors regressor:

```
Root mean squared error: 11502.36
Mean absolute error: 8866.14
Regression score: 0.05
```

All the others models in this conditions gave a negative result, here there is an example with SVR to see the result.

```
Root mean squared error: 11596.29
Mean absolute error: 9202.32
Regression score: 0.03
```

At this point the only way to try to have better results the augmentation was employed.
With the balanced augmentation there is a sensible improvement in the results:
With k-neighbours

```
Root mean squared error: 12076.47
Mean absolute error: 8042.31
Regression score: 0.28
```

With Adaboost

```
Root mean squared error: 10454.52
Mean absolute error: 8562.04
Regression score: 0.46
```

Anyway the magnitude of the prediction error is always similar.
By applying the more aggressive augmentation the results becomes even more stunning:
Adaboost

```
Root mean squared error: 7947.39
Mean absolute error: 6097.58
Regression score: 0.82
```

Anyway, as learned in the classification even if the augmentation does not replicate samples but generates new samples it is always easy to overfit. Again, by applying augmentation only on the train set the performance become again negative.

4 Conclusions

At the end of these analysis, there are a few considerations to express.
Both for the regression task and for the classification task, each possible approach did not bring any appreciable result.
From the classification side three sample for a class where too few for each possible effort to make the predictions better without causing an overfitting problem. In fact, the procedure of augmentation was not a rough approach, but a quite standard procedure to deal with unbalanced dataset. Usually, the oversampling is combined with another procedure called under sampling, in this case this procedure was unfeasible because it is impossible to train properly a model with only four samples for each class unless we have a very simple case in which they are sufficiently representative.
Moving on to the regression problem the situation was quite similar with a strongly uneven distribution of the samples. Without introducing ad hoc calculations to help the model to understand the function which associates input and outputs.
The procedure of oversampling always brought overfitting; this problem is probably related to the nature of the dataset. Generated samples were easier to understand by the models with the respect to the original ones.