Andrea Morelli 1845525

First Homework Machine Learning

Multi Uav Conflict Predictions

**Description of the homework**

The homework assigned has as objective a classification problem and a regression problem.

We are in an environment with 5 uavs and giiven the values of different sensors and other features we have to estimate in the classification problem the total number of conflicts between uavs and in the regression problem the minimum closest point of approach among all the possible pairs of uavs.

**Dataset**

The dataset is in the tsv format and is composed by the values of different sensors.

We have as input for each of the 5 uavs :

— the clockwise angle from north between the ith uav and its target (this value can assume values between o and 2 pi)

— the position components of the ith uavs in meteres

— the speed components of the ith uav in meters / seconds

— the position components of the ith uav in target in meters

**Data preprocessing**

I tried a lot of different thing in order to preprocess normalize and use the dataset.

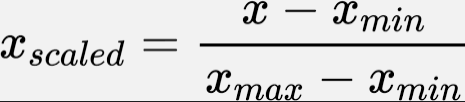
I noticed firstly that the clockwise angle could be normalized dividing it by 2 times pi in order to have values between 0 and 1 , but another approach could be applying the cosine function to the angle. This would allow to have smoother changes when switching between 0 and 2 pi in angles and the resulting values between -1 and 1 could be normalized to 0 and 1.

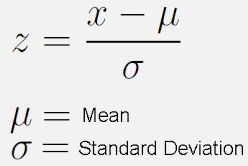
For all the other features a normalization with the min max normalizer or the standard normalization could be applied.

Instead of using the cosine directly on the angle i preferred to add new features to the dataset.I added the cosine of the angle as a new feature and the distance between the target and the uavs for every uav both on the x and y direction.

Adding those new columns would not help

On those new dataset created i would perform different normalizations

I tried the MinMaxScaler  with different features ranges for example 0 - 1 or -1 - 1.

I tried the StandardScaler  and this would normalize all the input values to have mean 0 and variance 1.

I tried the pca and the svd decomposition to get new features but found that they were not helpful

All those normalizations could be optimized with a grid search and a pipeline in order to see which performs better with the different models i did it for the support vector machine classifier.

I found that the standard scaler and the minmax scaler were performing better than the other methods in the classification task but also found that without normalization the performances of the different models were not that impacted.

**Imbalanced dataset in classification**

The problem of this dataset is that it’s highly unbalanced and the elements of the latest class do not have many instances.

That was a major problem when dealing with the classification.

I tried to use the random upsampler from the imblearn package, the smote method to upsample the minority classes and the method random downsample to try to deal with the majority class.

I tried different combinations of upsample lower classes and downsample most popular classes but all would perform poorly.

Without the upsampling the performances of the models were good just on the majority class so i would receive a good overall f1 micro score but all the other classes would not have good results so f1 macro would be bad but the weighted f1 would be better.

Sampling the training set and using it for the training of the models i would see not good results but the performances on the other classes were better (the f1 macro would be better but the f1 micro would decrease a lot).

Sampling before the split of the dataset would perform a lot better and the results would be optimal but would be like cheating: the distribution of the test set would be modified and the results would not be reflecting the real world so would not be able to generalize even if the results were a lot better.

**Imbalance dataset in regression**

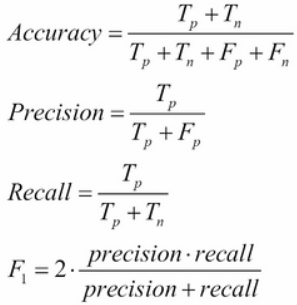
In the regression i found out that normalizing the dataset or not the performances would be pretty close but upsampling or downsampling would even worse than the results obtained.

**Evaluation procedure classification**

The dataset is splitted using the sklearn function in train and test set.

In the grid search i always used the k fold cross validation in the train set so i would keep the test set unwatched and the train set would be partitioned with k fold cross validation in train and validation set.

The metrics that were used or the classification were



the accuracy allows to see the number of correct predictions with respect to the total number of instances.

the precision allows to see how resistant the predictions are to the false positives.

the recall score allows to see how resistant the predictions are to the true negatives.

The f1 score is a combination of the above scores.

The f1 score has different version that have been used:

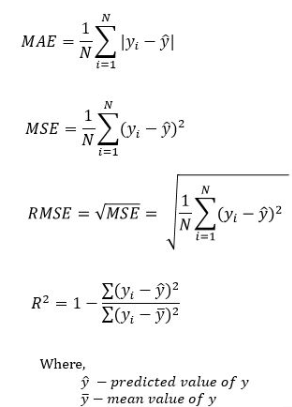
–the micro score calculates the f1 score on every predictions without taking in consideration the different classes (in this case it’s the same of the accuracy)

– the macro score calculates the f1 score on every classes and then takes the average of those scores

–the average score calculates the f1 score on every classes and weights everything with the number of elements for each class.

**evaluation procedure for the regression task**

The metrics that were used for the regression task were:



– the mean absolute error sees how much the predictions differ from the true value in absolute value in average

– the mean squared error sees how much the predictions differ from the true values squared

so its more sensible when we differ a lot from the true predictions.

–the root mean squared error is the root of the mean squared error

– the r2 score is a value with maximum of 1 and represents how good the predictions are

**different evaluations**

I divided the dataset in train and test set and i evaluated the performances on both of them.

The train set has been used to train the models and would perform better than the test set because all the elements have been seen during the training phase.

The train set is divided in train and evaluation set when the hyperparameters are optimized with the grid search or the random hyperparameters search.

**methods and algorithms used**

I used different models both for the classification task and for the regression task.

Classification:

The models that i tried for the classification were:

– decision tree

builds a tree like model that allows to classify every input following the branches of the tree until the leaves

– random forest

builds a collection of decision trees and uses them to classify an input instance

– adaboost

uses boosting to classify inputs giving more importance on inputs misclassified

– logistic regression

uses the logistic regression to classify input instances

–support vector machine

finds the best hyperplane to divide classes

– k neighbors

classifies the inputs finding the classes of the closest inputs

–gaussian naive bayes

assumes the inputs can be described by gaussians distributions and finds the best gaussians to approximate the inputs and uses them to classify new instances

–bagging

uses baggings to aggregate different predictions and output a class

– balanced bagging

bagging with random undersampling and oversampling with decision trees

–balanced random forest

random forest with random undersample and oversample

– dummy classifier

simple classifier that simply returns the most frequent class

Regression:

the models that i tried for the regression were:

– random forest

uses collection of decision trees in order to find the values of the min cpa

– adaboost

uses boosting to give more importance to inputs that have a score really far from the true value of min cpa

– linear regression

finds the best linear model to find the closest value to the min cpa

– support vector machine

uses svm regression with slack variables

– k neighbors

returns the closest or an average min cpa of the neighbors

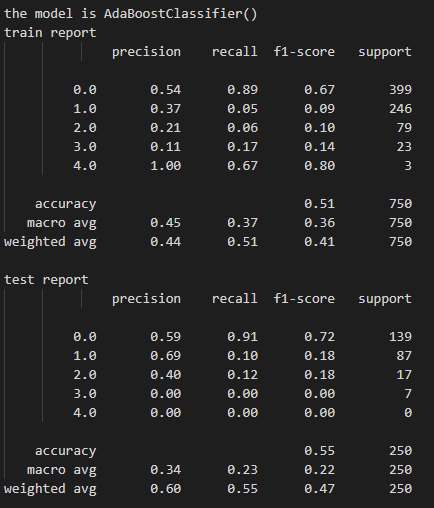
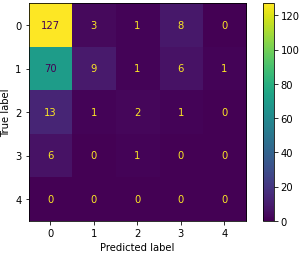
–decision tree

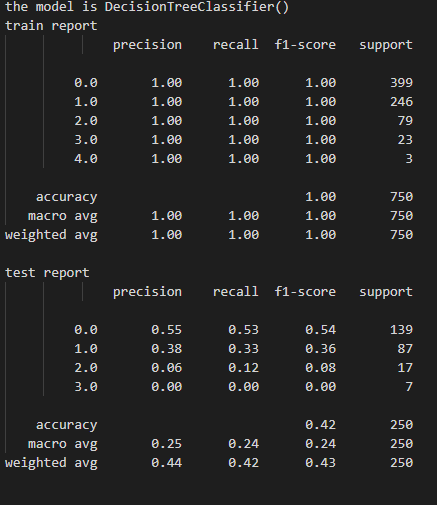
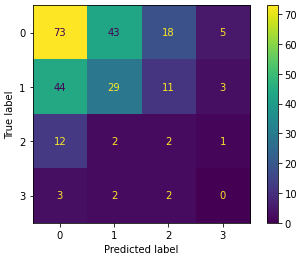
builds a tree with ranges of values and does something similar to a classifier to return a range where the regressed values is

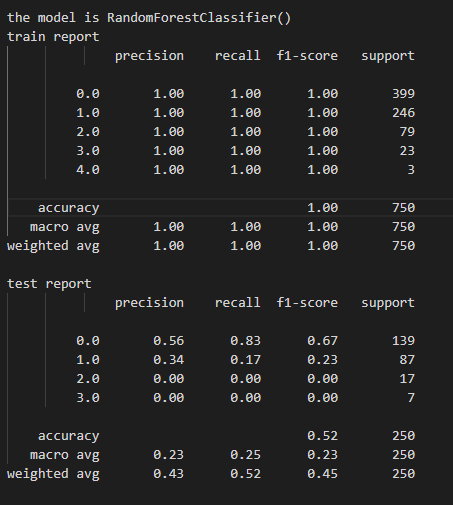
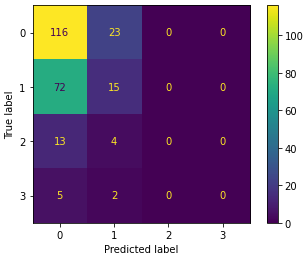
**configurations tried and best results**

Classification:

1)using the new features added(cosine of the angle and the distances from the target) ,normalizing the dataset without sampling we obtain the following results





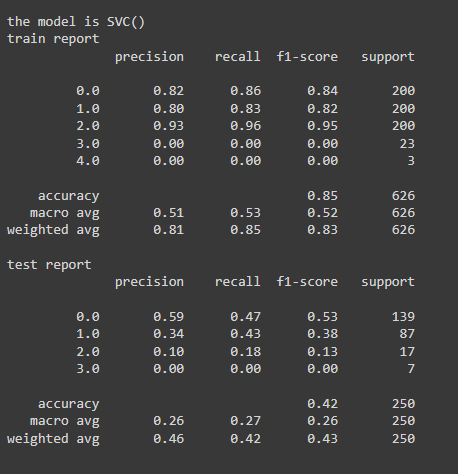
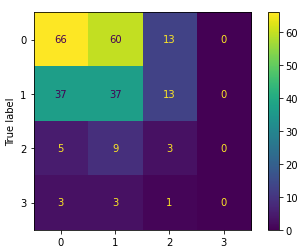


We can see that in general the f1 micro is good and also the f1 weighted but the macro is not. Thas because the model try to simply return the first class that is the most populated.

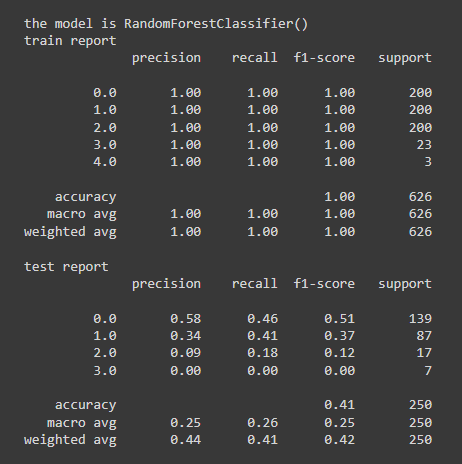
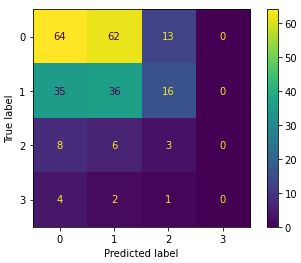
We can see also that the random forest and decision tree are overfitting a lot but pruning or building smaller trees or augmenting the number of trees in the forest would not help we would just see worse results both in the training and the testing set(i tried it in grid search)

2)Using downsampling and upsampling with both random and smote on the training set and minmax normalization

we balance elements ignoring the last 2 classes and having 200 elements for the first 3 classes using smote but the performances of the previous models were not good at all.Other models would perform in this way:



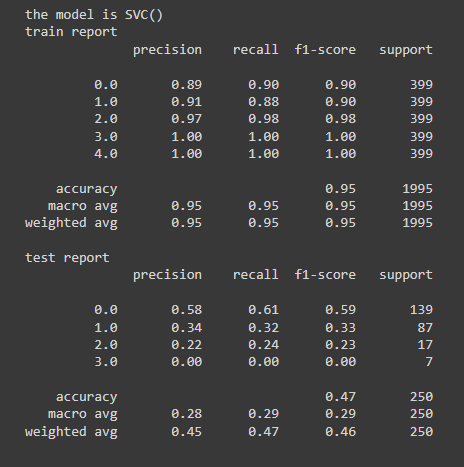
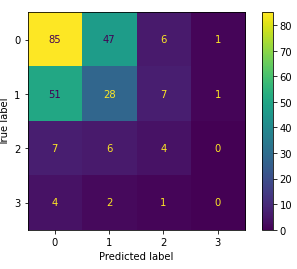
i show also the random forest for example to see the difference



we improve the performances on the other classes but the 0 class is not as accurate anymore( the micro is lower )

3) Balancing all the classes instead

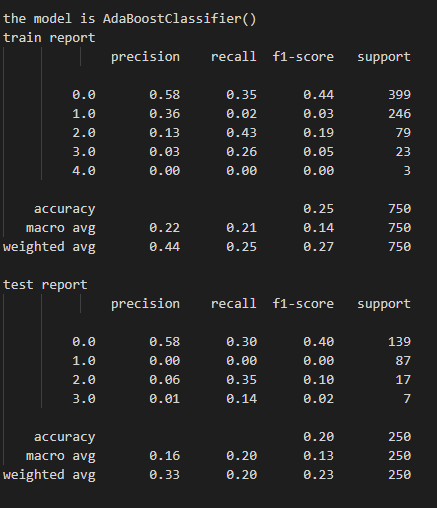
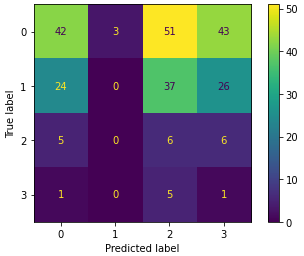
we can see an increased f1 macro with the svc



Using different normalizations would not improve results

4)Using svd or pca features would worse the performances of the models by a lot

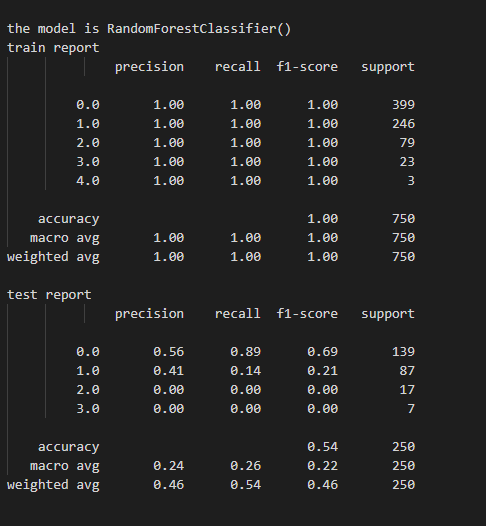
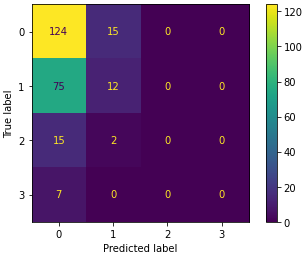
for example the adaboost with svd



adding my new features (the cosine of the angle and the distance between uavs and their target) would not improve the results

5) no normalization or upsampling would obtain worse results

for example the random forest would perform like this

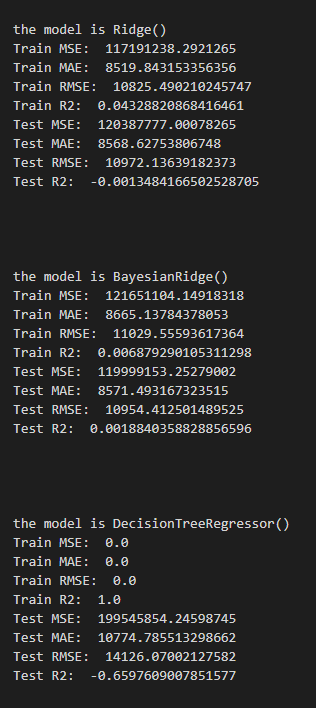
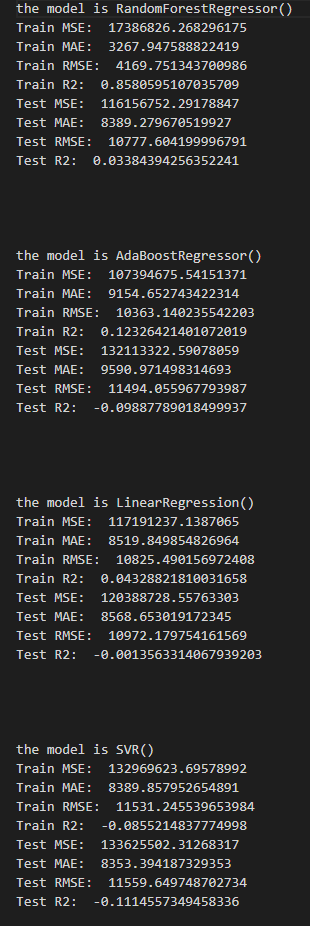


some models in general could be improved by reducing their “power” since they are overfitting completely the input but also doing that the performances would approximately be the same

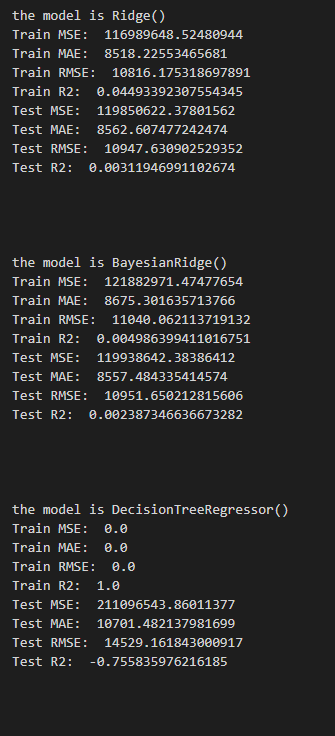
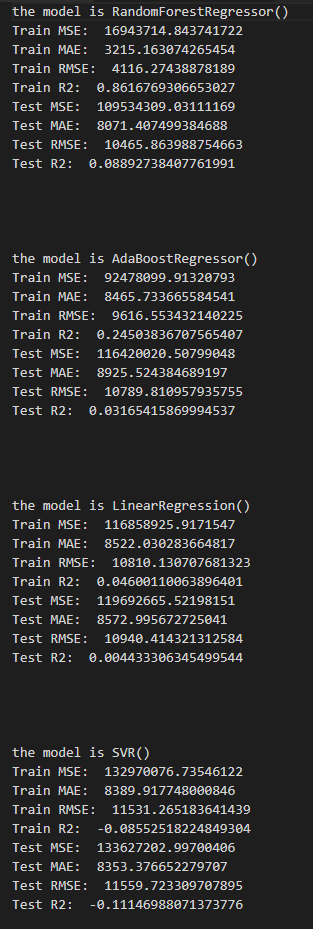
Regression:

for the regression task no matter how many different types of normalizations or hyperparameters for the models i would always obtain not good results

1)Not normalizing



2)Normalizing with minmax



the results would be slightly better buts still not good at all

and we can see that the models with the exception of the decision tree would underfit the data so more powerful models would have to be used

its obviously not useful to upsample and downsample the dataset to solve the regression task i tried it and obtained really poor results.

**hyperparameters tuning**

In order to find models with better results i started to do hyperparameters tuning.

The techniques applied were grid search and randomized search and those would find better hyperparameter for the models.

Grid search would have a systematic approach on finding hyperparameters for the models while the randomized search would try different hyperparameters randomly.

Classification hyperparameters tuning:

I applied different types of searches on different models

Tuning the hyperparameters without sampling would be possible with a simple grid search for example meanwhile tuning hyperparameters when applying sampling would require to use the imblearn package otherwise it will test the model just on the train set upsampled and the results would be biased.

Tuning without sampling:

I tuned different models one of them was the support vector machine classifier:

i tried different normalization methods like the minmax scaler and the standard scaler and different type of kernels (the rbf, the linear, the poly and the sigmoid optimizing the f1\_micro.

With standard scaler and sigmoid kernel i would obtain the best results using also k fold cross validation with 5 different partitions on the train set with a f1\_micro of 0.532 , applying the same model on the test set would produce an f1\_micro of 0.556.

Adding more parameters to tune would make the search a lot more time consuming.

Tuning with sampling:

The tuning when sampling has to be done with the imblearn pipeline .

This allows to not bias our results since we do not test on on the part of the dataset that has been upsampled so the results are not compromised.

The results are tho not really good

I tested the svc with parameters c kernel and class\_weight with different values and obtained a max f1\_macro score with c equal to 10 classweight to none and kernel rbf a value of 0.23 that is really low but its the correct way of doing hyperparmaeters search.

.

Using randomized search would not improve the performances but would be usually faster to evaluate

I also tried to reduce the power of decision trees and random forest since they were overfitting the dataset , pruning with a ccp\_alpah parameter max depth min\_ samples and min\_samples leaf but ht performances were not improving.

Regression hyperparameters tuning:

In the regression task i tried to do hyperparameters tuning but the performances would not improve

I tried ot do hyperparameter search on the random forest regressor.

This time there wasn’t the need to use imblearn and the pipeline since the sampling was not useful so i simply performed a grid search on n\_estimators and max\_depth .

**ways to improve**

We could use a more time consuming hyperparameters search but i don’t think the performances would improve a lot.

We could use different preprocessing for the data but also in this case i don’t think the performances would improve since i tried to use pytorch with neural network and saw that also in this case the performances were not improving so the features that we are provided with are ok and could not be improved by non linearities.

I think the main problem of this dataset is that it is too small and trying to balance it or making it bigger with simple sampling would not be good and have valuable information added.

The thing that maybe could be working is the use of models that try to perform regression and classification together for example using neural network in order to have a composed loss and have one task regularize the other and viceversa.

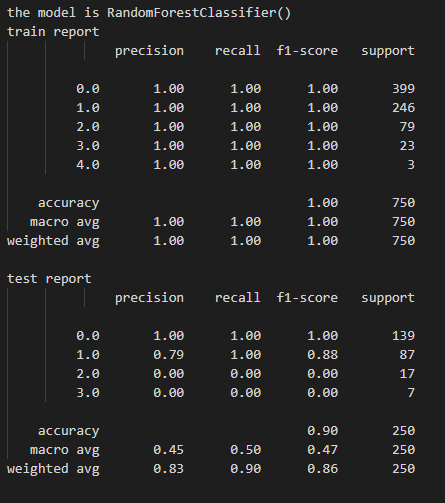
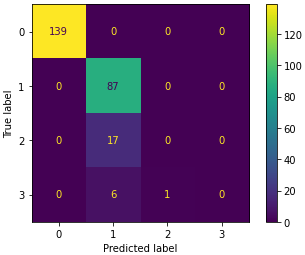
If that doesn’t improve the performances i thing the only thing that could work is have a bigger dataset or extensive hyperparameters search.

**things found and what could be done**

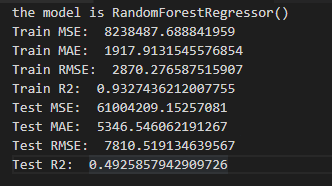
The thing i found out is that as one could expect min cpa and number of collisions to be related so solving one task could allow to solve the other.

So i used the min cpa column in the training set to solve the classification task with incredible results and the viceversa using the number of collisions to solve the regression optimally.

random forest classiication



random forest regression



Obviously this is not something that could be done to solve this homework

If we could use the other column of the y set to solve the tasks i would use it on the training set and then on the test set i would inset the value of the closest element in the train set for example using cosine similarity or the output of another model.

With neural network we could solve the tasks together and using a combined loss to regularize the model (restricts the values the parameters can have so its like we have less parameters so its regularizing) and i think we could obtain better results.