MULTI-UAV CONFLICT RISK ANALYSIS

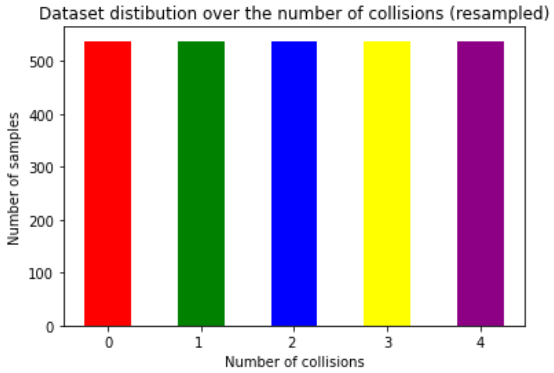
Data Preprocessing

Starting from the data preprocessing, the technique that has been applied is the min max scaler. For each feature of the dataset, every sample has been scaled with the following method:

$$x_{scaled} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

Where xmin and xmax is the feature range over all the sample in the dataset.
This preprocessing data has been applied over all the feature both for the classification and for the regression task.
In the regression task the preprocessing has been applied also for the true value given.

Another preprocessing that has been made is the resampling of the elements in the dataset. All the classes have been balanced respect to the number of samples in the dataset.



The new distribution of the sample that has been used to do some test with some model covered in the next chapters. With this new distribution, there are 538 samples for each class.

The technique used to resampling is the Syntentic Minority Oversampling Technique. (SMOTE)
This method uses a KNN mechanism on the real existing sample to add new ones in the classes under sampled.
Has been used the parameter K equal to 2.
It is worth noticing that this technique can be useful in dataset in which the sample of the different class are more or less linearly separable (there can be some outliers but the main distribution of the classes do not overlap); this observation is not valid for this dataset because the data are not linearly separable (it's possible to see it in a plot in the multidimensional reduction analysis).

Although this property of the dataset, the balanced dataset has been used in some test to check the behavior of the models with augmented data.

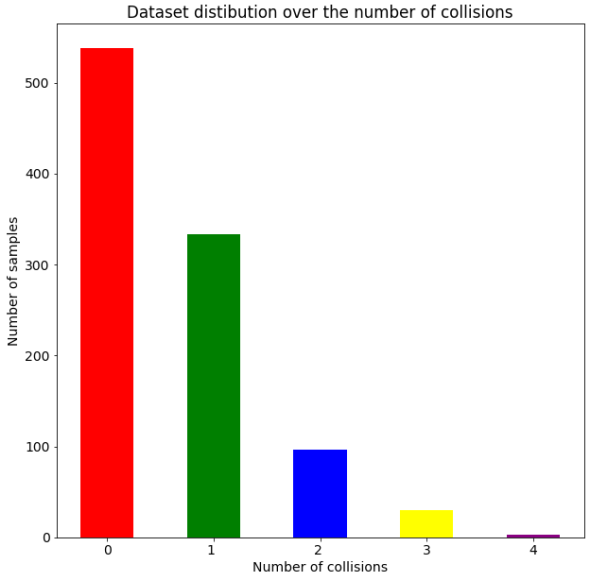
Note that resample the train set dataset after splitting it (in train= 66% and test set) is not possible. The reason is related to the size of the class '4' in the training set: it contains too few neighbors to create a new sample of the same class with the SMOTE technique.
For this reason the dataset has been resampled on the entire dataset.

The last consideration about the resample is that, according to the domain, the new sample may not have sense in the real word, but again, the balanced dataset has been used in some test to

check the behavior of the models with augmented data, and in any case all test have been done on the original normalized test set even if training happened with balanced normalized train set.

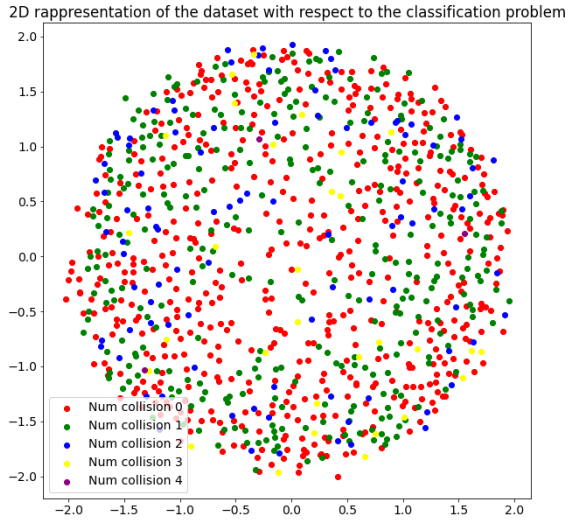
Distribution of the dataset & Multidimensional reduction Analysis

On the left the histogram that show how the original data (normalized) are unbalanced spread along the classes of the classification problem.

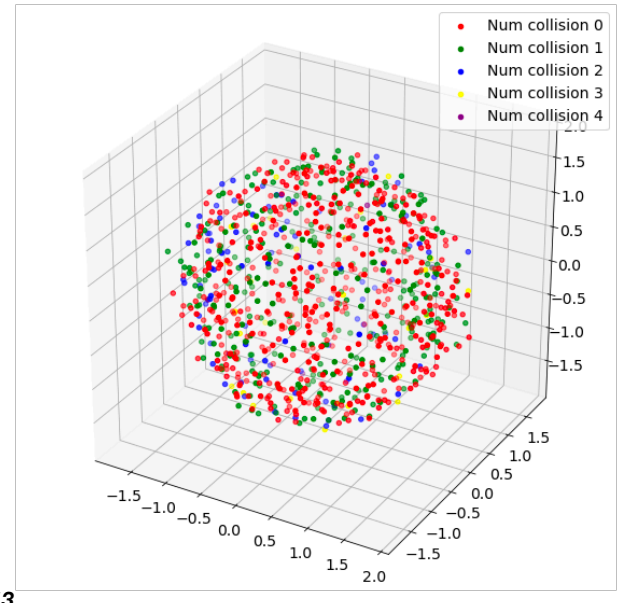


Looking at the distribution is expected that the model performs better on classifying samples for which has been trained more by the dataset: in this case the class 0 contains more sample then the others, so it should be the one on which the model should performs better.

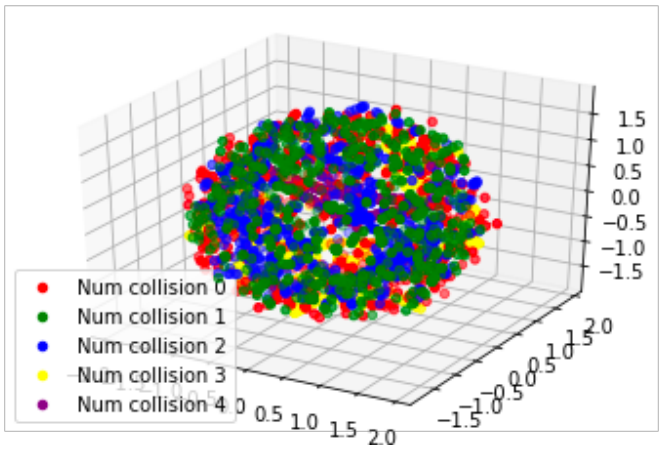
Another idea that has been tried to exploit consist in a multidimensional reduction analysis of the feature space in two and in three dimensions for classification and in two and one dimension for regression. The main idea was validate that the data in the dataset are not linearly separable even in two and three dimensions.
Here the plots about the distribution of the data for the classification problem both for unbalanced and balanced dataset:



Unbalanced Dataset: 2D reduction Classification



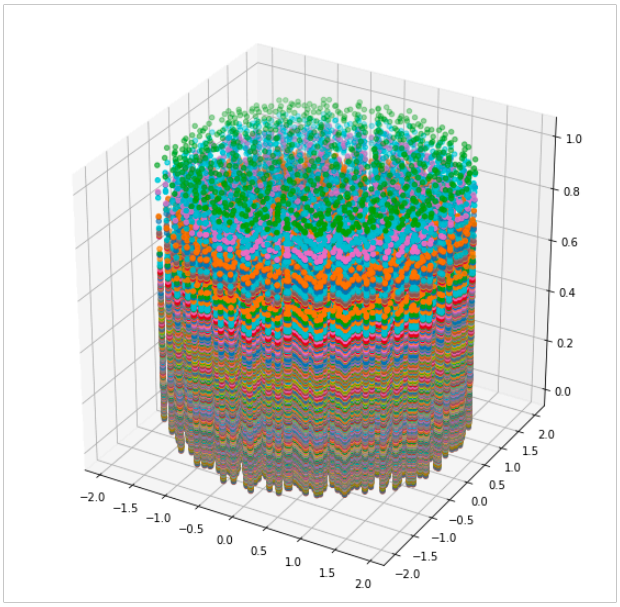
Unbalanced Dataset: 3D reduction Classification

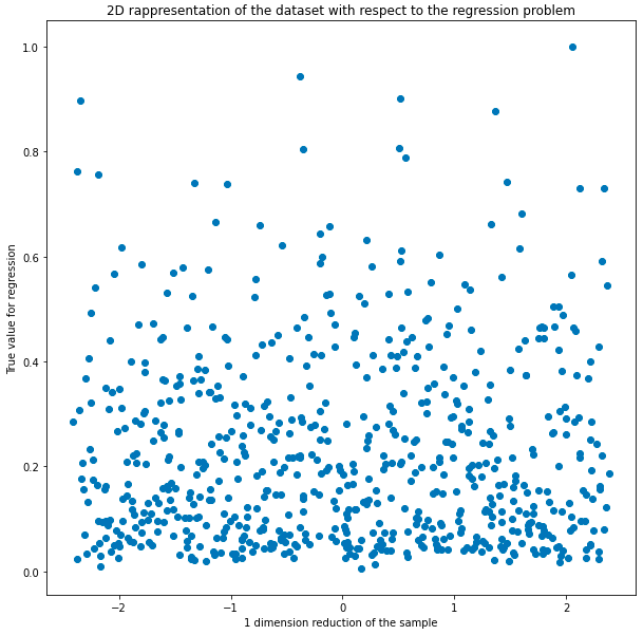


In all the cases is easy to observe that the data are more or less equally distributed in the space and they overlap too much.

Unbalanced Dataset: 3D reduction Classification

Here the plots about the distribution of the data for the regression problem in 2 and 3 dimensions:

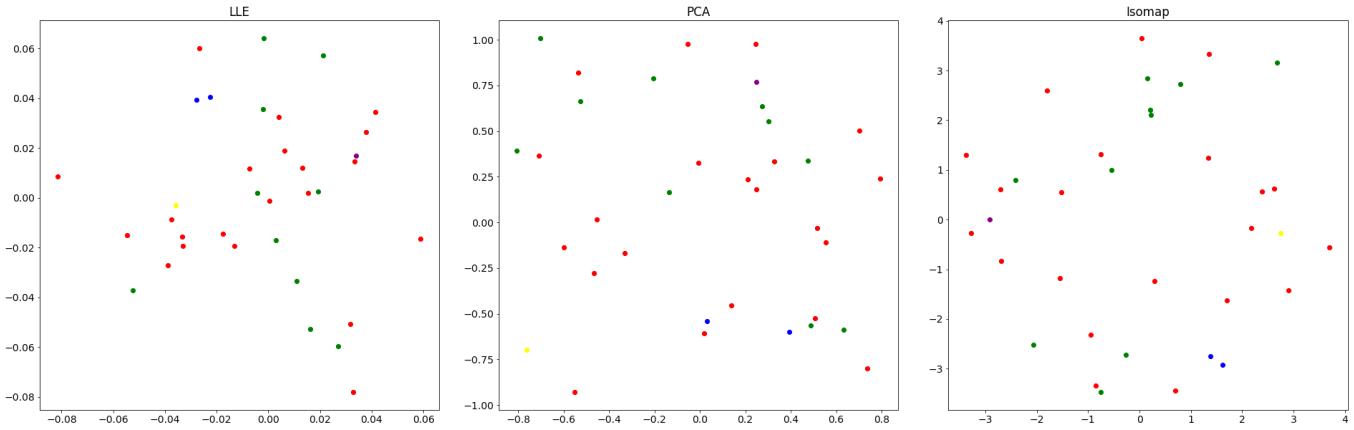




Note that for the first plot, the z axes represent the true value of the regression problem, while in the second plot is the y axes .
Also in this case, the multidimensional reduction doesn't not present data in a separable way.

Not other analysis about dimensionality reduction has been made, but the expectation is that the data will be not separable in an easy way.

Other Multidimensional Reduction



In any of this cases the data are presented in a separable form, so this analysis gave no advanced on which data to train the models on.

Methods for classification

Data division

The dataset has been sliced using the sklearn function train_test_split with 66% of sample used for training.
So for the unbalanced normalized dataset 660 elements for training and 340 for the test and for the balanced normalized dataset 1775 elements for training.

Logistic Regression

Description

The first model used is the logistic regression with an hyper parameter optimization over the penalty and the regularization term.
The multi class parameter has been setted with value ‘ovr’ (one versus rest), in other words for each classes has been fitted a binary problem.
This are the chosen possible value for the hyper-parameters:

Parameter name	Values
Penalty	L2 , none
C	20 values in logarithmic space (log space (-4,4))

For the parameter ‘solver’ has been provided only one possibility: the newton-cg.
With a cross validation parameter equal 10, the grid search has fitted the model totally 400 times : 10 folds for each of 40 candidates.

This model has been fitted both on the normalized unbalanced dataset and on the balanced one.

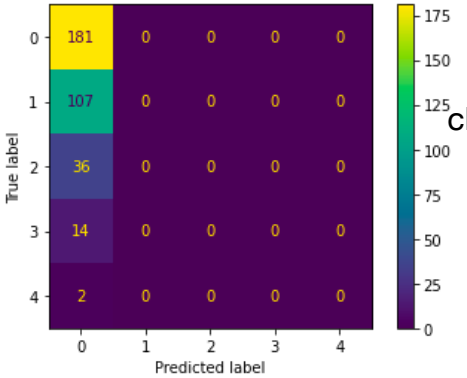
Results

For the unbalanced dataset this is the report for the best classifier found.
The hyper-parameters that gives the best score are:

'classifier__C': 0.0001, 'classifier__penalty': 'l2'

	Precision	Recall	F1-Score	Support
0 class	0.53	1.0	0.69	181
1 class	0.0	0.0	0.0	107
2 class	0.0	0.0	0.0	36
3 classs	0.0	0.0	0.0	14
4 class	0.0	0.0	0.0	2
Accuracy			0.53	340
Macro Avg	0.11	0.20	0.14	340
Weighted Avg	0.28	0.53	0.37	340

The confusion matrix is the follow one:

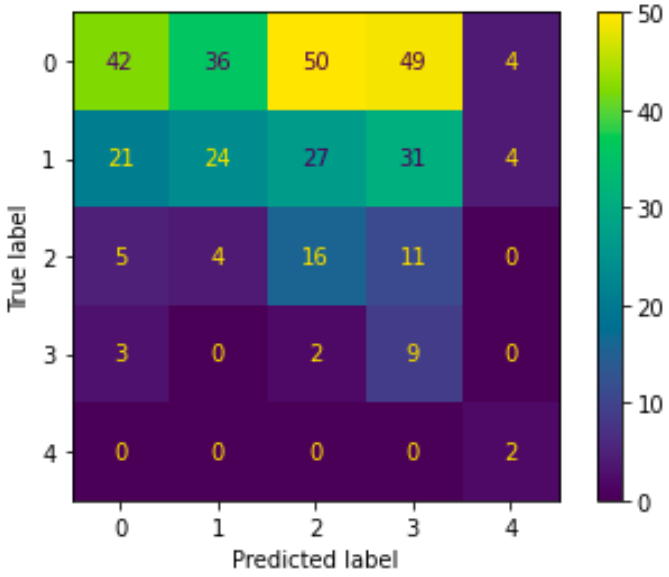


From the matrix it's possible to observe that the accuracy of the model is influenced from the score values of the class zero.

Furthermore the model is not even able to classify neither one of the sample with classes different from zero.

The second test of fitting has been done on the balanced dataset with the same configuration of the hyper-parameter for the grid search as before.
Here the results are as follow.

	Precision	Recall	F1-Score	Support
0 class	0.59	0.23	0.33	181
1 class	0.38	0.22	0.28	107
2 class	0.17	0.44	0.24	36
3 classs	0.09	0.64	0.16	14
4 class	0.20	1.00	0.33	2
Accuracy			0.27	340
Macro Avg	0.28	0.51	0.27	340
Weighted Avg	0.46	0.27	0.30	340



From the confusion matrix it's possible to see that this model start to increase performance with respect to all the other classes different from 0 but the accuracy on the overall test set is less then the accuracy of the previous model.

Note that the time for the training on the unbalanced dataset is about 25 seconds while for training on the unbalanced dataset is about 1 minute and 6 seconds.

SVM for classification

Description

For this method the idea was exploit the power of the model, working on the unbalanced dataset trying two different hyperparameter optimization: the grid search and the bayesian optimizer.
Then with the Grid Search has been applied also ensemble methods to compare which one give the highest accuracy.

Grid Search HPO

This are the chosen possible value for the parameters:

Parameter name	Values
Gamma	1, 0.1, 0.001
C	5 values in logarithmic space (log space (-4,4))
Kernel	Rbf, poly, sigmoid, linear

With a cross validation parameter equal 10, the grid search has fitted the model totally 600 times : 10 folds for each of 60 candidates.
The hyperparameter that gives the best score are:

'classifier__C': 0.0001, 'classifier__gamma': ‘1’, ‘classifier__kernel’: ‘rbf’

The result is exactly like the logistic regression: all the sample are classified with class 0.
The time for this training is much more then the logistic regression: 53 minutes.

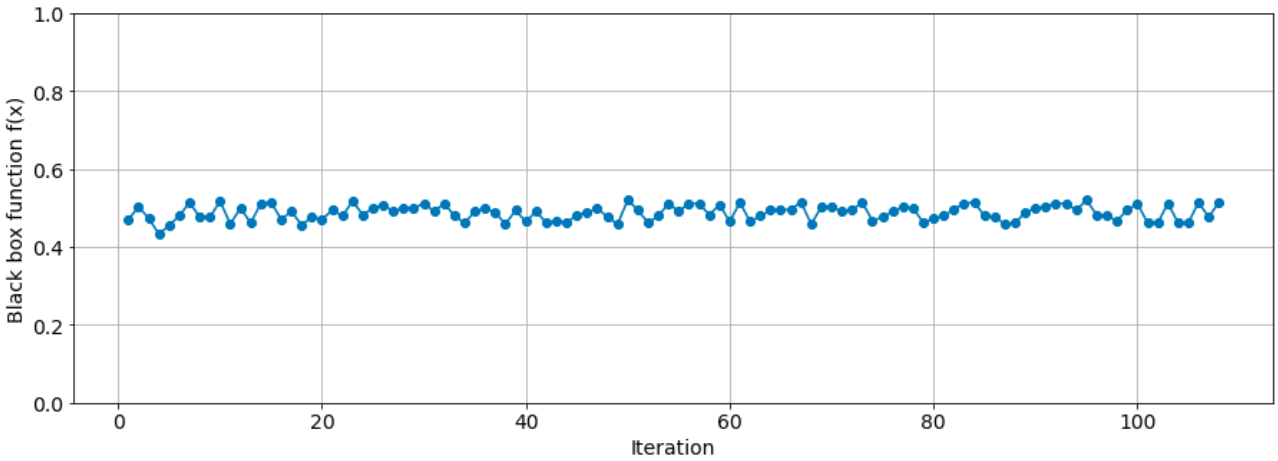
Bayesian Optimizer
Given from the gridsearch that the best kernel to use is the rbf, in this optimization method the hyperparameter search is expressed as bounds as follow:

'C': from 1.e-04 to 1.e+04, 'gamma': from 0.001 to 1

With 100 iteration (starting from 8 initialization points) for the process of optimization, the time required that 53 seconds, the best hyperparameters found are

‘C': 2692.73 , 'gamma': 0.99 with the best accuracy value of 0.52

This is the plot of the accuracy during the iterations.



Ensemble Grid Search
For bagging the two hyper-parameters and their possible value are the following:
‘Max sample’ : 5 random value in a range of 5, length of train set divided by 2
‘Number of estimators’: 10, 20, 30

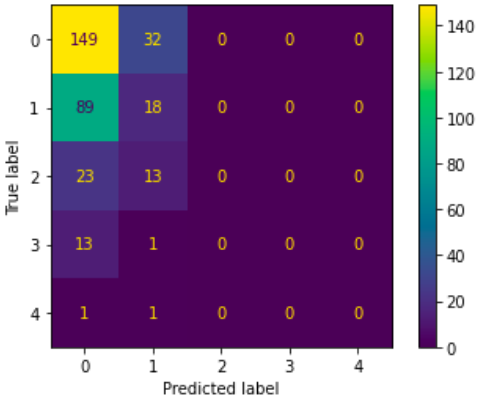
For boosting parameters:

'Learning rate' : 5 random value from an uniform distribution over the range [0.0 , 1)
'Number of estimators': 10, 20, 30
The booster method used is the AdaBoost.

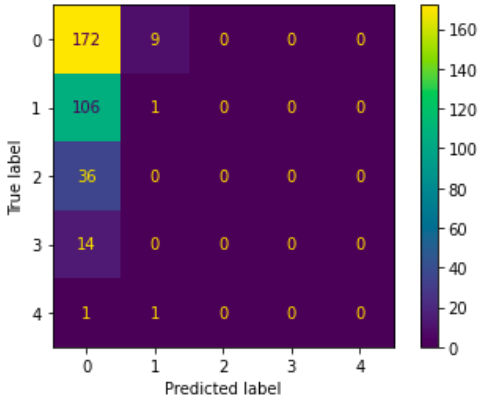
Here are the results:
Fit Grid Search on Bagging took 1.3 minutes, with AdaBoost just 9 seconds.
The best configuration of the hyper-parameters and the accuracy:

Bagging :
'max_sample' : 73, 'n_estimators' : 20
accuracy : 0.5227

Boosting:
'learning_rate': 0.288, 'n_estimators': 10
accuracy: 0.5379



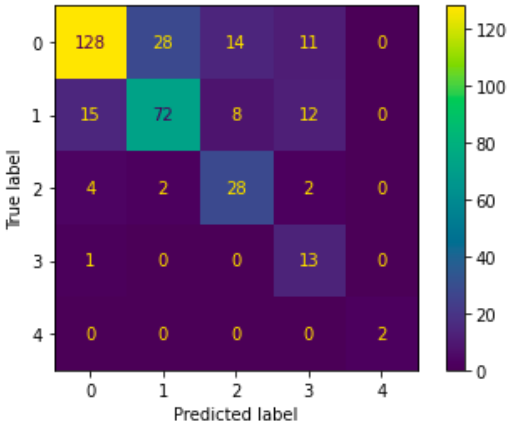
Grid Search Bagging SVC



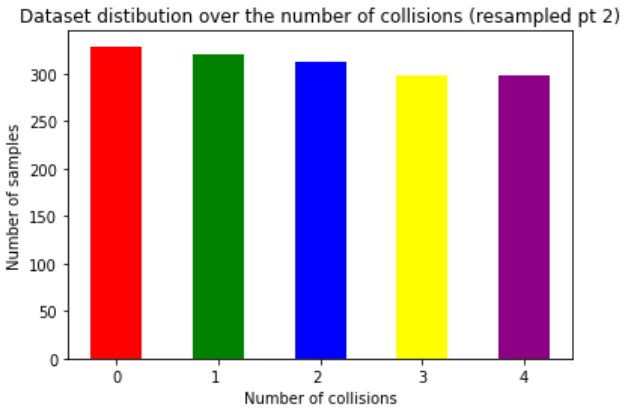
Grid Search Boost SVC

Decision Tree & Random Forest for classification

Description
The model has been trained on the balanced normalized dataset with max_depth equal 8 and gave an accuracy score of 0.35 on the unbalanced normalized train set with the following results.
Here are presented two fits done with balanced training set: have been removed from balanced training set all the sample that were also in the test set of the original dataset. Accordingly to this there was a slightly different distribution over the balanced classes.

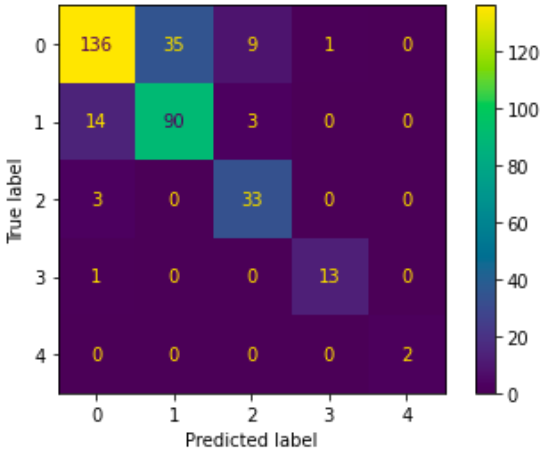


Conf .Matrix Decision tree trained on balanced data and tested on original test set



New distribution slightly different from the perfect balanced one

The other test that has been made was with Ensemble for decision tree: random forest. With random forest have been made different tests, it gives really high measure of performance, with tree of max_depth = 10 and 10 estimators it reaches 0.81 as accuracy score.



Conf. Matrix of Random forest trained on balanced dataset

	Precision	Recall	F1-Score	Support
0 class	0.88	0.75	0.81	181
1 class	0.72	0.84	0.78	107
2 class	0.73	0.92	0.81	36
3 classs	0.93	0.93	0.93	14
4 class	1.00	1.00	1.00	2
Accuracy			0.81	340

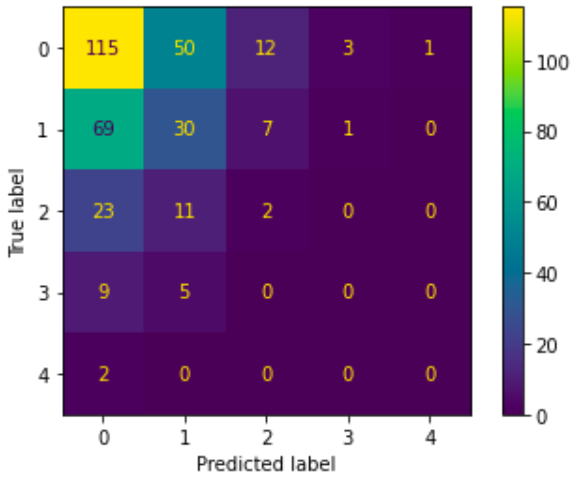
	Precision	Recall	F1-Score	Support
Macro Avg	0.85	0.89	0.87	340
Weighted Avg	0.82	0.82	0.81	340

XGBoost

Description

This is another model that has been tested on classification on unbalanced normalized dataset. It has been used with the following parameters:
n_estimators=2, max_depth=6, learning_rate=0.001, objective=multi:softmax.

The accuracy Given from this model is 0.43 with the following confusion matrix.



Also this model is not able to classify correctly samples of class 2,3 and 4 and the accuracy is being influenced the most just from class 0.

Conclusions about classification

Here Is presented a summary of the performance about the different previously discussed models.

Model	Dataset	Ensemble	Hyper-params optimizer	Accuracy	Time
Logistic Regression	Unbalanced normalized		Grid Search 400 fits	0.53	25 s
Logistic Regression	Unbalanced normalized		Grid Search 400 fits	0.27	1 m 6 sec
SVM	Unbalanced normalized		Grid Search 600 fits	0.53	53 m
SVM	Unbalanced normalized		Bayesian Optimizer	0.52	53 sec
SVM	Unbalanced normalized	Bagging	Grid Search	0.5227	1.3 m

Model	Dataset	Ensemble	Hyper-params optimizer	Accuracy	Time
SVM	Unbalanced normalized	Boosting	Grid Search	0.5379	
DT	Balanced normalized			0.35	15 sec
DT	Balanced normalized	Random Forest		0.82	
XG Boost	Unbalanced normalized			0.43	

Note that in this case the model that performs better is the Random forest.

Note also that the time measure is not available for all the test and furthermore it is a very variable measure accordingly to the resources given in google Colab, or in general from the hardware infrastructure.

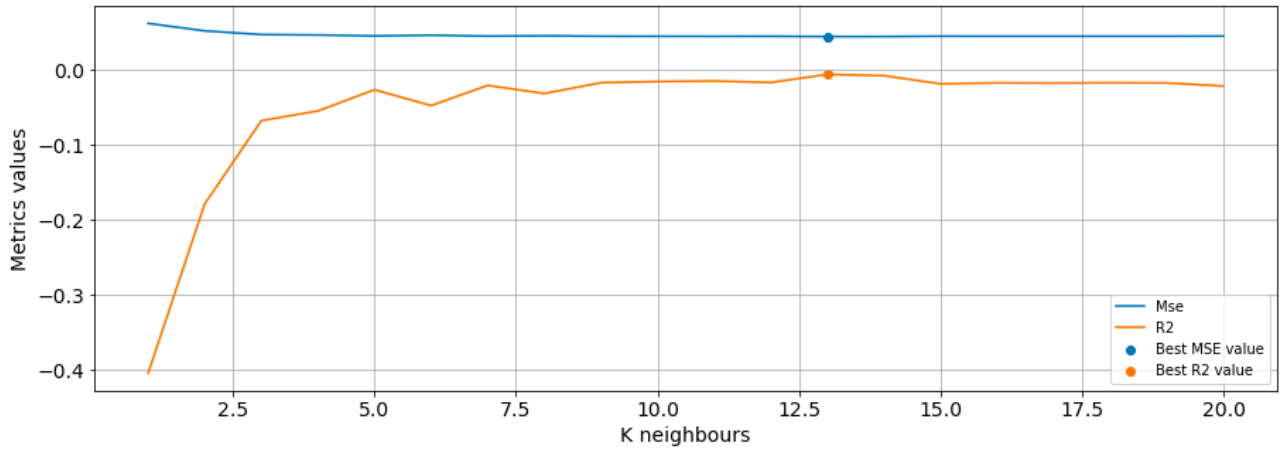
Methods for regression

KNN for regression

Description

Has been tested KNN in a range of K from 1 to 20 to find the one with the best r2 and the best MSE.

The data used in every iteration is the training set normalized
This are the results obtained.



The bests result appears to be in the same Iteration: K=13 with MSE 0.044 and R2 -0.006

SVM for regression

Description

With SVR have been made the following test:
- a grid search with cross validation = 5 (300 fits) and the unbalanced normalized dataset over the following parameters:
    1. Kernel: ref, poly, sigmoid, linear
    2. Gamma: 1, 0.1 , 0.001
    3. C: 5 values in logspace with the range -4, 4

The metrics that have been observed in this section are the mean squared error and the R2 score.

The time to train the first grid search has been of 47 minutes

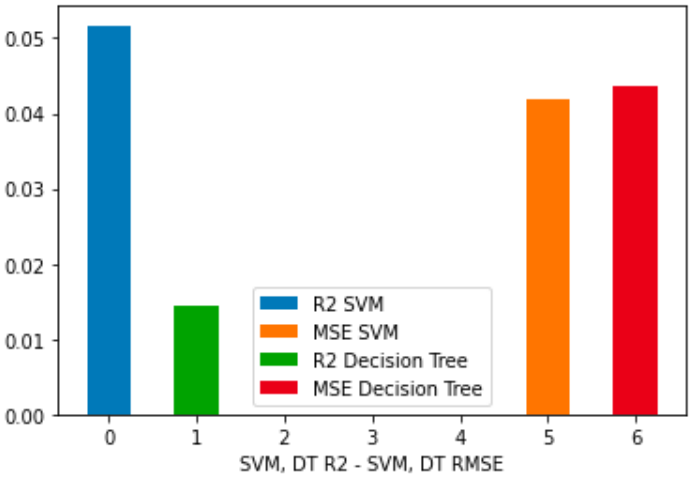
Results

The best result for parameters is:
'C': 1.0 , 'gamma': 1, 'kernel': rbf
MSE score : 0.032

Svm Baseline and Random forest for Regression

Description

In this section have been tested the models, with default parameters given from sklearn: Svm for regression and Random Forest.



Baseline Svm Vs Random Forest

Both have been trained and tested on the unbalanced normalized dataset.

Grid Search Svm Ensembles

Description

In this section have been tested:
- grid search bagging over KFold with 5 as number of split
- grid search boosting KFold with 5 as number of split

Both this models have been trained and tested on the unbalanced normalized dataset.

The number of estimators provided as parameter were for both a list of 5 random values in a range 5,50: 36, 10, 41, 24, 24.

Grid search Bagging
Search parameter: 'max_samples': list of 5 values [144, 170, 188, 22, 219]

Best parameters found: 'max_samples': 219, 'n_estimators': 41

The method took 1min 24s seconds to be trained and got MSE 0.0318 from the best estimator on test set.

Grid search Boosting (Adaboost)
Parameters: 'learning_rate': [0.1, 0.01, 0.001]

Best parameters found: 'learning_rate': 0.001, 'n_estimators': 24

The method took 8 min 38s seconds to be trained and got MSE 0.0391 from the best estimator on test set.

Conclusions about regression
Here Is presented a summary of the performance about the different previously discussed models.

Model	Dataset	Ensemble	Hyper-params optimizer	R2	MSE	Time
KNN	Unbalanced normalized		Sequential search over K	-0,006	0,044	
SVM	Unbalanced normalized			0,051	0,041	
SVM	Unbalanced normalized	Bagging	Grid Search		0,029	1m 24s
SVM	Unbalanced normalized	Boosting	Grid Search		0,043	8m 38s
DT	Unbalanced normalized	Random Forest		0.0086	0,04	

Ignoring the R2 score because the model are too naive for this task, the one to focus on is the MSE: the best model looks to be the SVM trained on the unbalanced normalized dataset and optimized with grid search.