Homework 1 - Multi-UAV conflict risk analysis

Alessandro Garbetta, 1785139

March 26, 2024

1 Introduction

This homework deals with the development and implentation of a multi-classifier for detecting the number of collision between UAV (Unmanned Aerial Veichle) and a regression problem on the predict the minimum Closest Point of Approach (CPA) among all the possible pairs of drones. Our informations are on initial position, target position, velocity and orientation of the aerial vehicles.

The Unmanned Aerial Veichles knwon also as drone, are aircraft without any human pilot or passengers n board. The use of UAVs can be applied in many different fields from military to recreational. In our case we see how different autonomous drones can cooperate in different jobs, and even interact with human.

In our Homework we will work on data of fly of five UAVs.
We were provided with a dataset "train_set.tsv", a pandas dataframe, i.e. a tab-separated file that contains 1000 samples. The dataset contains various information: orientation between the drone and its target, positions of the vehicle, instantaneously velocity, target position and finally the number of collision and the minimum closest point of approach for each of five drones.

UAV_5_track	UAV_5_x	UAV_5_y	UAV_5_vx	UAV_5_vy	UAV_5_target_x	UAV_5_target_y	num_collisions	min_CPA
1.607546768	-80967.56143	37684.58241	213.1754418	-7.837820472	41342.56704	33187.60649	3	1673.734894
4.315805787	13285.31165	-41245.54575	-196.9787579	-82.48907782	-2806.112817	-47984.17472	0	51230.54779
0.9547955866	21679.56331	-34471.15445	208.5807259	147.6519215	78980.50969	6091.533217	0	18668.17777

Figure 1: Example of sample of the 5-th UAV

To reach the goal of our classifier, we will use as information: each data of the drones and the num_collisions(we have a minum of zero collisions, and a maximum of 4 collisions) which will be the label of our classifier. For the regression, we use the same data but instead of num_collisions we use min_CPA as regression prediction. The provided dataset is ready to use, but we have to do a normalization of the data. There is one problem, it is not balanced.
After the dataset loading and manipulation phase, a model was created to train the multi-classifier to find the number of collisions in the various lines of code. Various models were tested (also modifyng the dataset to resolve the problem of imbalnced dataset), observing the results with the help of evaluation techniques such as confusion matrix; also for regression problem

2 Load of dataset

First we loaded the dataset, using the pandas library, we read the tsv file, and we split the various information using the tab separator. We extracted all the colums: "UVA_i_x", "UVA_i_y", "UVA_i_x", "UVA_i_x", "UVA_i_x", "UVA_i_x" and "UVA_i_x" for each UAV and also the "n_collisions" and "min_CPA".

3 Normalization

The sklearn module has effective methods for data preprocessing and other machine learning tools. We see differents types of normalization.
Before being able to split the data set, an operation of normalization was done on the data. The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information.
Normalization is also required for some algorithms to model the data correctly. For machine learning, every dataset does not require normalization. It is required only when features have different ranges. This operation creates a sparse matrix with normalization data that can change with different parameters.

We have mainly considered 3 types of normalization present in the sklearn.preprocessing library, and they are: the normalize, the minmax_scale and the StandardScaler.

- normalize – scale input vectors individually to unit norm (vector length). We can choose different parameters to norm.

  - L1: $z = \|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |\boldsymbol{x}_i|$
  - L2: $z = \|\boldsymbol{x}\|_2 = \sqrt{\sum_{i=1}^{n} \boldsymbol{x}_i^2}$
  - Max: $z = \|\boldsymbol{x}\|_\infty = \max |\boldsymbol{x}_i|$

  Figure 2: Possible parameters to norm

- StandardScaler – standardize features by removing the mean and scaling to unit variance. The standard score of a sample x is calculated as:

$$z = \frac{x - u}{s}$$

  Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set.

- minmax_scale – transforms features by scaling each feature to a given range. This estimator scales and translates each feature individually such that it is in the given range on the training set. The transformation is given by:

$$X\_std = \frac{X - X.min(axis = 0)}{X.max(axis = 0) - X.min(axis = 0)}$$

$$X\_scaled = X\_std(max - min) + min$$

where min, max = feature_range. This transformation is often used as an alternative to zero mean, unit variance scaling.

Not one of the normalization techniques was favored, but they were tried with the various models. Comparisons between the various models and normalization will be made later.

4 Split of Dataset

One mistake that can be made is to not divide the dataset with the input samples into two subsets, the training dataset and the testing dataset. In fact, the testing of the model cannot be done on the same elements that have been used for the training. There are standard ratios for dividing the dataset, usually it is recommended to use 2/3 of the dataset for the training set, and the remaining 1/3 for the test set. In order to divide the dataset it has been used the function train_test_split of the library sklearn.model_selection. Among the required parameters there is the test_size, that is in percentage the size of the future test set, and then, for complementarity the train set.

5 Performance Metrics in Classification

Before treating the fitting and the evaluation of the model, we expose the metrics that have been used in order to study which was the best predictive model.

- Accuracy - is one of the values that in a more immediate way gives a complete vision of the model and its operations. Unfortunately this parameter does not give detailed information about the specific problems, this lose information about the prediction of the various classes.

$$error\_rate = \frac{\#errors}{\#instances} = \frac{FN + FP}{TP + TN + FP + FN}$$

$$accuracy = 1 - error\_rate = \frac{TP + TN}{TP + TN + FP + FN}$$

  Furthermore, this is not a good metric when datasets are unbalanced, so we prefer to use other metrics.

- Recall - it helps when the value of false negatives is high

$$recall = \frac{truepositive}{realpositive} = \frac{TP}{TP + FN}$$

- Precision - it helps when the value of false positives is high

$$precision = \frac{truepositive}{predictedpositive} = \frac{TP}{TP + FP}$$

- f1-score - this parameter improves the accuracy by combining precision and recall. In this way it is possible to have a low number of false positives and false negatives,

calculating the true values of the model, not being disturbed by false alarms. f1-score is a decimal value between 1 and 0, indicating maximum and minimum operation respectively.

$$f1 - score = \frac{truepositive}{predictedpositive} = 2 * \frac{precision * recall}{precision + recall}$$

Finally, the confusion matrix was used: each column represents the predicted values, while each row represents the true values, the element on row i and column j identifies the number of cases in which the classifier classified the "true" class i as j. In binary classification the 4*4 matrix that will be constructed in our case, represents in position [1,1] the true negatives, in position [1,2] the false positives, in position [2,1] the false negatives and in position [2,2] the true positives.
A good functioning of the model is identified in the matrix when the values in the diagonal are high and the values outside the diagonal are very low, tending to zero.

A multi-class classification problem can be split into multiple binary classification datasets and train a binary classification model each. One-vs-rest (OvR) is a heuristic method for using binary classification algorithms for multi-class classification. We evaluate each class versus the orhers like they are only one.

One-vs-One (OvO) is another heuristic method for using binary classification algorithms for multi-class classification, but this time we evaluate each class versus each other class.

6 Performance Metrics in Regression

Before treating the fitting and the evaluation of the model, we expose the metrics that have been used in order to study which was the best predictive model.
Once we have our model fitted, we can get the results to check whether the model works satisfactorily and to interpret it.

- Coefficient of determination – tells us which amount of variation in y can be explained by the dependence on x, using the particular regression model. A larger R^2 indicates a better fit and means that the model can better explain the variation of the output with different inputs. Also, the attributes of model are .intercept_, which represents the coefficient b, and .coef_, which represents b:

  The value of b illustrates that our model predicts those response when x is zero. The value b means that the predicted response rises by those when x is increased by one.

- Mean Squared Error – or MSE, it is also an important loss function for algorithms fit or optimized using the least squares framing of a regression problem. It minimizing the mean squared error between predictions and expected values. The MSE is calculated as the mean or average of the squared differences between predicted and expected target values in a dataset.

$$MSE = \frac{1}{N * \sum_i^N (y\_i – ypred\_i)^2}$$

Where y_i is the i'th expected value in the dataset and ypred_i is the i'th predicted value. The difference between these two values is squared, which has the effect of removing the sign, resulting in a positive error value.

- Root Mean Squared Error – or RMSE is an extension of the mean sqaured error. It's calculated the square root of the error, which means that the units of the RMSE are the same as the original units of the target value that is being predicted.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N * \sum_i^N (y\_i – ypred\_i)^2}}$$

- Mean Absolute Error – or MAE, like RMSE the units of the error score match the units of the target value that is being predicted, but unlike the changes in this metric are linear and intuitive. The MAE does not give more or less weight to different types of errors and instead the scores increase linearly with increases in error. It is calculated as the average of the absolute error values.

$$MAE = \frac{1}{N * \sum_i^N (|y\_i – ypred\_i)^2|}$$

7 Modelling and Fitting - Classification Problem

The biggest problem with this task was working with unbalanced datasets, so the performance of the models was first observed with the provided dataset produce results very similar. Then an attempt was made to use a model with parameters suitable for the problem, without significant improvements. Improvements were only made by going to operate on the dataset and oversampling the data. In this way we made the data distribution homogeneous and final values were better. Several models were tried, and the ones that performed best were the the DecisionTreeClassifier, and the RandomForestClassifier.

- DecisionTreeClassifier - As the name suggests, a decision tree model partitions the data by making decisions based on the answers to a set of questions. Based on the characteristics of the training data set, the decision tree model learns a set of questions to determine (via inference) the labels of the sample classes.
  Using the decision algorithm, we start at the root of the tree and partition the data according to the feature that yields the maximum information gain, i.e., we need to define an objective function that we want to optimize with the tree learning algorithm. The information gain is simply the difference between the impurity of the parent node and the sum of the impurities of the child nodes: the lower the impurity of the child nodes, the higher the information gain. However, for simplicity and to reduce the combinatorial search space, most libraries (including scikit-learn) implement binary decision trees. This means that two child nodes Dleft and Dright depend on each parent node.
  We must be careful: the deeper the decision tree, the more complex the decision boundary becomes, resulting in an overfitting problem. Using scikit-learn, one can restrict the decision tree to a maximum depth, say 3, by specifying it in the method parameters. In our case, we did not impose any depth limit. One can also specify a parameter defining the impurity criterion (by default Gini impurity). Intuitively, the Gini impurity can be considered as a criterion to minimize the probability of

misclassification.
Advantage: decision trees are particularly interesting if we are interested in interpretability.

- RandomForestClassifier - It generates multiple trees but always starting from the same training set, choosing however the optimal split in a random subset of the possible splits. In this way each tree is different.
  Random forests have gained great popularity in machine learning applications during the last decade due to their good classification performance, scalability and ease of use.
  The idea behind using a set of learning systems is to combine multiple weak learning systems to build a more robust model, a strong learning system that offers better generalization error and is less susceptible to overfitting problems.
  Advantage: we do not have to worry too much about choosing a good value for the hyperparameters. In general, we do not have to prune the random forest, since as a whole the model is quite noise resistant compared to the individual decision trees. The only parameter we really need to worry about, in practice, is the number of k trees we chose for the random forest. Typically, the larger the number of trees, the better the performance of the random forest classifier will be, with the disadvantage of increased computational cost. Fortunately, we don't have to build the random forest classifier manually using individual decision trees; scikit-learn already provides a ready-to-use implementation.

8 Modelling and Fitting - Regression Problem

In this work we use a (Multiple) Linear Regression model and Polynomial Regression model.

- Linear Regression - there is some dependent variable y on the set of independent variables x = (x, . . . , x), where r is the number of predictors, you assume a linear relationship between y and x. This relationship is the regression equation and there are regression coefficients and random error. Linear regression calculates the estimators of the regression coefficients or simply the predicted weights. These estimators define the estimated regression function f(x). This function should capture the dependencies between the inputs and output sufficiently well. Regression is about determining the best predicted weights—that is, the weights corresponding to the smallest residuals. To get the best weights, you usually minimize the sum of squared residuals (SSR) for all observations.

- Polynomial Regression - now, we assume polynomial dependence between the output and inputs. In addition to linear terms like in previous case, now the regression function f(x) includes non linear terms.

9 Evaluation - Classification

Once the model fitting phase is finished, the prediction of the test set is done. For every case it has been visioned the confusion matrix.
With unbalanced dataset the result are very poor, and the correct classification is only

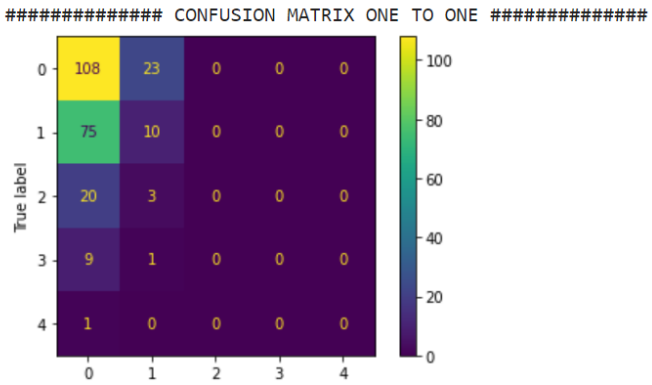

Figure 3: Confusion Matrix and metrics of RandomForestClassifier with unbalanced dataset

	precision	recall	f1-score	support
0	0.51	0.82	0.63	131
1	0.27	0.12	0.16	85
2	0.00	0.00	0.00	23
3	0.00	0.00	0.00	10
4	0.00	0.00	0.00	1
accuracy			0.47	250
macro avg	0.16	0.19	0.16	250
weighted avg	0.36	0.47	0.38	250

Figure 4: Metrics of RandomForestClassifier with unbalanced dataset

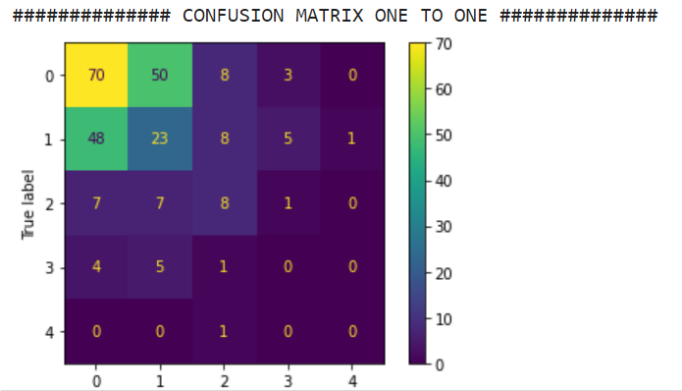

Figure 5: Confusion Matrix DecisionTreeClassifier with sample_weights

	precision	recall	f1-score	support
0	0.54	0.53	0.54	131
1	0.27	0.27	0.27	85
2	0.31	0.35	0.33	23
3	0.00	0.00	0.00	10
4	0.00	0.00	0.00	1
accuracy			0.40	250
macro avg	0.22	0.23	0.23	250
weighted avg	0.40	0.40	0.40	250

Figure 6: Metric DecisionTreeClassifier with sample_weights

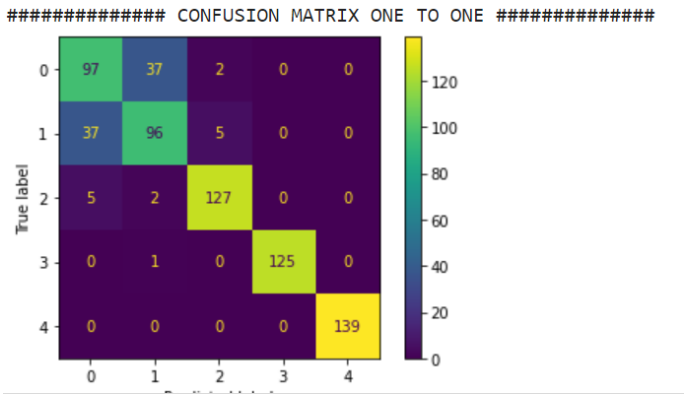

Figure 7: Confusion Matrix DecisionTreeClassifier with sample_weights

	precision	recall	f1-score	support
0	0.70	0.71	0.71	136
1	0.71	0.70	0.70	138
2	0.95	0.95	0.95	134
3	1.00	0.99	1.00	126
4	1.00	1.00	1.00	139
accuracy			0.87	673
macro avg	0.87	0.87	0.87	673
weighted avg	0.87	0.87	0.87	673

Figure 8: Metric DecisionTreeClassifier with sample_weights

with the class that are more popular in our data, while for the minor class representation the values are near zero.

We tried using the decision tree algorithm with the sample_weights parameter to mitigate the effects of data imbalance. There was an improvement in the least classes, albeit minimal. We can see the Confusion Matrix in Figure 5 and the others metrics in Figure 6. After this evaluation we use the function SMOTE of the sklearn library to perform an oversampled of the dataset; a method used when the sample of different class are not homogeneous. After that, we try to use different classification algorithm. When we use RandomForestClassification with new balanced dataset we use n_estimators (that indicates the number of trees in the forest) egual to 200 and we obtain a better result of DecisionTree (Figure 9).
Increasing the number of trees worsens the execution time of the model, so we kept the estimator value equal to 200 (trying with 500 or 1000 and the result are similars).

For completeness we point out that tests were also done with other models, including BernoulliNP, and the results were not very good. For this reason we have concentrated on the above models.

Models test with balanced data were also evaluated using K-fold CV, which is a cross-validation method. The set is shuffled randomly (or by setting a size, and this time 0.33 was entered), the data set is divided into k groups (15).
Each group is used as a validation set and all others as train sets. The model is established using the train set and evaluated using the validation set, the model evaluation score is stored in a list.
Each result was equivalent to those obtained with the confusion matrices

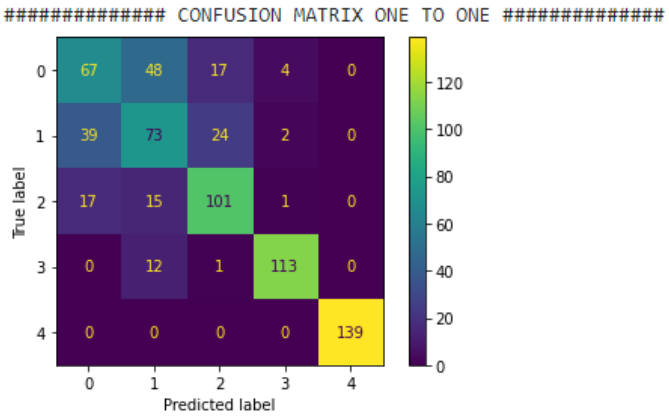

Figure 9: Confusion Matrix DecisionTreeClassifier with oversampled dataset

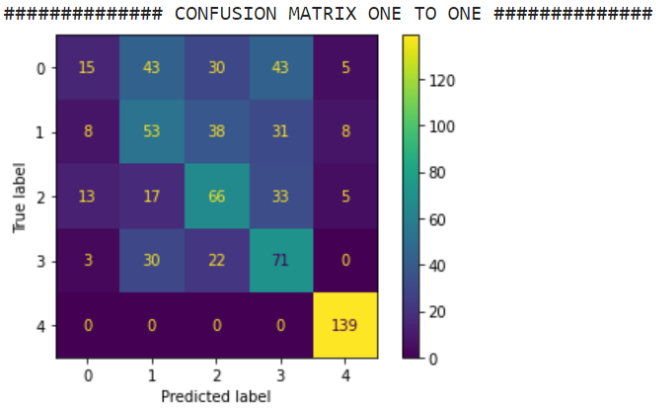

Figure 10: Confusion Matrix Bernoulli with oversampled dataset

	precision	recall	f1-score	support
0	0.38	0.11	0.17	136
1	0.37	0.38	0.38	138
2	0.42	0.49	0.46	134
3	0.40	0.56	0.47	126
4	0.89	1.00	0.94	139
accuracy			0.51	673
macro avg	0.49	0.51	0.48	673
weighted avg	0.50	0.51	0.48	673

Figure 11: Metric Bernoulli with oversampled dataset

10 Evaluation - Regression

Also for the regression problem we tried different types of normalize, as for the classification problem. In this case we decided to show the results obtained mainly with "normalize" and with "StandardScaler". By using the two types of normalization with the linear regression model, we can see that the coefficient of determination remains very similar, but by using the normalize the values of the error types examined decrease. We can observe the result in Figure 11 and Figure 12.

By using a nonlinear model,but polynomial, the results in terms of coefficient of determination and errors definitely improve. Again using the "normalize" function

```
coefficient of determination: 0.037166
intercept: 0.06979297255679504
slope: [-3.53496077e+02  1.92882804e-02 -2.43872045e-04 -3.90363243e+00
  2.73030395e-02 -1.28360212e-02  7.68550628e-03 -8.60970093e+01
  2.88094600e-02 -2.24352762e-03  6.10781007e+00 -6.92708874e+00
 -2.28045423e-02  9.48286279e-03  5.76975789e+02 -1.70354714e-02
  3.20790052e-03 -2.03226287e+00 -1.93836026e+00  1.95284389e-02
  2.84232995e-04 -1.24748623e+02 -1.03774334e-02  8.15836373e-03
 -1.30603127e+00 -9.58923160e-01 -9.48029374e-03 -6.10687982e-03
 -3.17316645e+02 -2.22952729e-03 -2.81146128e-04  3.09408912e-01
  3.82787745e-01 -1.10261489e-02  3.33241799e-04]
MSE 0.0026324780754452153
RMSE 0.05130768047227642
MAE 0.040655281838334285
```

Figure 12: Linear Regression with normalization: normalize

```
coefficient of determination: 0.039889
intercept: -4.153473004894509e-17
slope: [-0.05136293  0.0463816   0.01712845 -0.08348408  0.01561483 -0.03858463
  0.03444627 -0.01524692  0.09582771 -0.00156839  0.09049437 -0.10544194
 -0.10025413  0.04759789  0.10600815 -0.08637347  0.00990499 -0.04462734
 -0.0242237   0.11210935 -0.00887856 -0.01448322 -0.04162017  0.03462693
 -0.01405391 -0.00652111 -0.04963918 -0.03765215 -0.06028161 -0.01885123
  0.00398805 -0.01696564  0.02391659 -0.04050371 -0.01301264]
MSE 0.960110778356408
RMSE 0.9798524268257991
MAE 0.7713133897021257
```

Figure 13: Linear Regression with normalization: StandardScaler

to normalize the data, they improved, the coefficient increased, and the errors, in a different way, decreased. We can observe the result in Figure 13 and Figure 14.

```
coefficient of determination: 0.9696338441681647
intercept: 4.572034836277517
MSE 8.302388490369815e-05
RMSE 0.009111744339241425
MAE 0.00725070458097281
```

Figure 14: Polinomial Regression with normalization: normalize

```
coefficient of determination: 0.7820090298454634
intercept: -808445939.6871395
MSE 0.2179909701545365
RMSE 0.4668950311949534
MAE 0.3698381443694556
```

Figure 15: Polinomial Regression with normalization: StandardScaler

11 Conclusion

In conclusion, we can say that in the first problem, the classification problem, the most complicated problem was working with an unbalanced dataset, which did not bring much improvement during model evaluations. Only with an intervention on the samples, increasing their number, and distribution, better results could be obtained. However, because the minority class is still made up of a limited amount of unique data points, the model is susceptible to memorising data patterns and overfitting.
In the regression task, on the other hand, what brought most differentiation in the results was the type of normalization used. In addition, a nonlinear regression brought improvements.