Classification and Regression for UAVs

Simone Bodi, 1815606

University La Sapienza

Abstract

The aim of the work has been to perform a classification and regression task. To solve the classification problem it has been necessary to find a model that learns how to estimate the total number of conflicts among UAVs. To solve the regression problem it has been necessary to find a model that learns how to predict the minimum Closest Point of Approach (CPA).

1. The Dataset

The dataset consists of a series of samples, where for each sample the features are:

- UAV_i_track;
- UAV_i_x and UAV_i_y;
- UAV_i_vx and UAV_i_vy;
- UAV_i_target_x, UAV_i_traget_y.

where i is the the number of i-th drone. For each row of the dataset we have five drones, associated with each row we also have the respective output, the number of collisions for the classification task and the Minimum Closet Point for the regression task. In our case the dataset is unbalanced, this means that the target variable has more observations in one specific class than in the others. Furthermore, our dataset has features with different units of measure: the angle features are represented in radian , the position features are represented in meters, the velocity features in meters/second, so we need to normalize the data.

2. Classification Task

Our aim in this task has been to train a model to recognize the number of collisions, from 0 to 4, having the trajectories, speeds, angles of 5 drones. So by formalizing the machine learning model we have had to create a model able to approximate a function:

$$f : X -> Y \tag{1}$$

where $X \subseteq R^{35}$ and 35 is the number of features, $Y = \{0, 1, 2, 3, 4\}$ is the set of possible collisions.

2.1. Normalize the dataset

First the dataset is read through the **pandas** library. Then the dataset is divided in two parts: X which contains the 1000 rows with all features from 1 to 35; Y which contains the output associated with the 1000 rows of X. Since the dataset wasn't normalized, it has been normalized. Normalization allows the features values to be expressed using the same scale. L2 normalization is used, which consists of taking each value and dividing it by the square root of the sum of all squared values. More precisely the formula for $x \in X$ is:

$$x\_norm = \frac{x}{\sqrt{\sum_{i=1}^{n} i^2}} \tag{2}$$

where $i \in X$ and $n = |X|$. It has been necessary to normalize only the X set. Then it has been necessary to split the sets X and y into training set and test set.

2.2. Split the Dataset

To train and test the model, the train_test_split sklearn function has been used, which generates four sets: a set X_train to train the model, the respective y_train, the set X_test to test the model and the respective y_test. With a size of 33% for testing and 67% for training. It has been chosen a random state with a value of 7 that decides the shuffle of data for splitting the datasets randomly.

2.3. Balance the dataset

At first we had an unbalanced dataset that had 538 elements for class 0, 333 elements for class 1, 96 elements for class 2, 30 elements for class 3 and 3 elements for class 4 (Figure 1). To

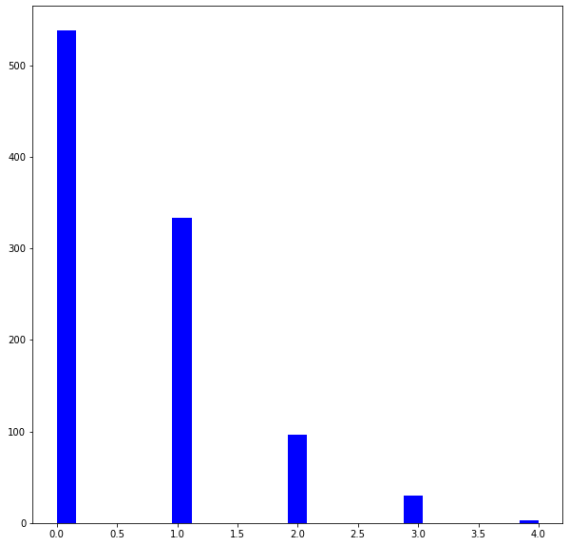

Figure 1. The target class distirbution before the balancing

balance the dataset the Undersampling technique taken from the library of imblearn has been used, it consists of randomly eliminating data from the class with the largest number of samples, or of eliminating a certain number of elements for the classes you choose, this is done with the sampling strategy parameter, where you can indicate for each class how many samples you intend to eliminate. Overuse of this technique can lead to excessively decreasing the model's ability to classify elements. Various tests have been carried out by decreasing the number of elements in the majority classes. It has been decided to apply undersampling only on the training set, this to train the model not in an unbalanced way, in fact if we had left the original number of samples for class zero, the model would have had greater accuracy but only because it correctly classified all

elements of class zero, but not correctly classifying all the others. The elements of class zero have been decreased from 538 to 220. The decrease in the number of samples for the majority class has allowed to obtain a slightly higher f1-score also for class 1, which did not happen with oversampling technique. After balancing the training set, the distribution of the target classes changes like this:

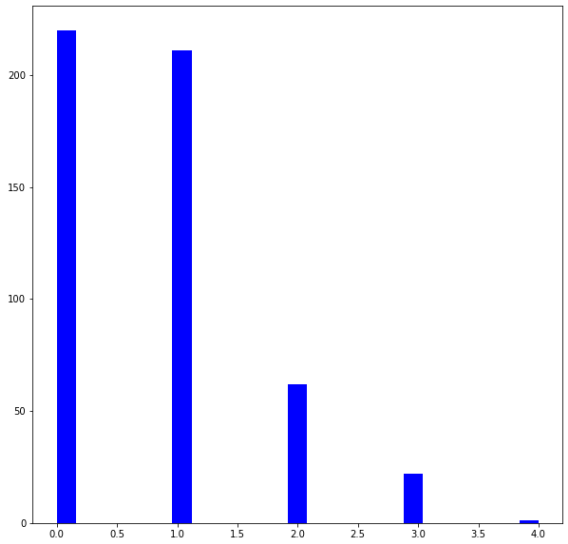

Figure 2. The target class distirbution after the balancing

2.4. The models and the setup

Different models have been used for the tests: Bernoulli Naive Bayes, Gaussian Naive Bayes , Random Forest Classifier, Gradient Boosting Classifier, Perceptron, SVM. The models that have given the best results have been Random Forest Classifier and Support Vector Machine. The Random Forest Classifier it is based on the Decision Tree framework, so we start from the root of the tree and taking a series of binary decisions we arrive at a leaf node that represents the classification output . Moreover the Ensamble learning has been used, which is a process where multiple models are used, the results of the models are averaged, and the one that outputs the best classification result is found. The Random Forest algorithm combines ensemble learning methods with the decision tree framework to create multiple randomly drawn decision trees from the data, averaging the results to output a result that frequently leads to strong predictions/classifications. Grid search has also been used; it is an optimization technique that attempts to compute optimal values of hyperparameters. It is an exhaustive search that is performed on the specific parameter values of a model. The model is also known as an estimator. In particular, the various parameters on which the grid search worked have been:

- n_estimators: the number of decision trees you will be running in the model;
- max_depth: this sets the maximum possible depth of each tree;
- min_samples_split: The minimum number of samples required to split an internal node;
- min_samples_leaf: The minimum number of samples required to be at a leaf node.

Support vector machine has also been used extensively in testing, which is the usually preferred method for classification, essentially SVM finds a hyperplane that creates a boundary between data types. For multi-class problems a binary classifier is created for each class. The result of each class can be either the data point belonging to that class or the data point not belonging to that class. Also for the SVM we have used the grid search which tested the following parameters:

- The Kernel: Radial Basis Function Kernel, polynomial, sigmoid, linear;
- The Gamma, which decides how far the influence of a single training example goes during the transformation, which in turn affects how tightly decision boundaries end up surrounding points in the input space. If there is a small gamma value, the more distant points are considered similar. So more points are clustered together and have smoother (perhaps less accurate) decision boundaries. Larger gamma values cause points to be closer together (may cause overfitting). In particular, the following values were tested: 0.1, 1.0, 10;
- The C parameter, which controls the amount of smoothing applied to the data. Large values of C mean low regularization which in turn causes the training data to fit very well (may cause overfitting). Lower values of C mean more regularization which causes the model to be more tolerant of errors (may lead to lower accuracy).In particular, the following values were tested: 0, 0.1, 0.2, 2, 10.

2.5. The test and Performance evaluation

The tests we have carried out have led to the following results:

- Bernoulli Naive Bayes, accuracy: 49%
- Gaussian Naive Bayes, accuracy: 47%
- Random Forest Classifier, accuracy: 54%
- Gradient Boosting Classifier, accuracy 50%
- Perceptron, accuracy: 45%
- Support Vector Machine: 55%.

Analyzing the models, the Support Vector Machine has been the best model for this expirement. SVM was tested with two technique of preprocessing: Oversampling, Undersampling.

On the test with Undersampling, it has been decreased the number of samples of majoirty class from 538 to 220, it has been obtained a general accuracy with K-Fold of 55% with +/- 0.05 but for X_test it has been obtained an accuracy of 45%, with a f1-score of 50% for 0-class and 44% for 1-class.

The SVM with Undersampling technique gave the best results indeed. An accuracy of 55% has been reached but unfortunately the training model isn't generalized enough. Analyzing the models, the Support Vector Machine gave the best results indeed, on the test set we have obtained: An accuracy of 55% has been reached but unfortunately the model has not correctly classified the classes 0,1,2,3,4, this is because the number of samples within the training and test set is higher for the class zero (figure 3). Observing the confusion matrix we can see that for class zero we had about 50% decision-making capacity, for class 1 the model had indecision but in any case the number of correctly classified was greater than those classified inadequately, for all the other classes instead a good classification was not obtained. To analyze how the model behaves on the whole dataset, the K-Fold validation technique has been used, which returned an accuracy of 55% with a standard deviation of 0.05%, this is not a very good result, it could be improved by increasing the number of samples for the minority classes but this would risk leading to overfitting, having a higher accuracy on the test, but a very low accuracy after the K-Fold validation.

2.6. Conclusion

In this classification work, the aim has been to find a model that best classifies the number of possible collisions related to the trajectories of 5 drones.

First of all, the dataset has been preprocessed as each row presented values expressed in different scales, normalizing the data to ensure that the model could regulate learning and improve classification capacity.

```
test size 0.333
random state 7
              precision    recall  f1-score   support

           0       0.50      0.50      0.50       167
           1       0.38      0.52      0.44       122
           2       0.00      0.00      0.00        34
           3       0.00      0.00      0.00         8
           4       0.00      0.00      0.00         2

    accuracy                           0.44       333
   macro avg       0.18      0.20      0.19       333
weighted avg       0.39      0.44      0.41       333

[[83 84  0  0  0]
 [58 64  0  0  0]
 [19 15  0  0  0]
 [ 5  3  0  0  0]
 [ 1  1  0  0  0]]
[0.52552553 0.58858859 0.55555556 0.55855856 0.51951952]
Accuracy: 0.550 (+/- 0.05)
```

Figure 3. The result of model on test set

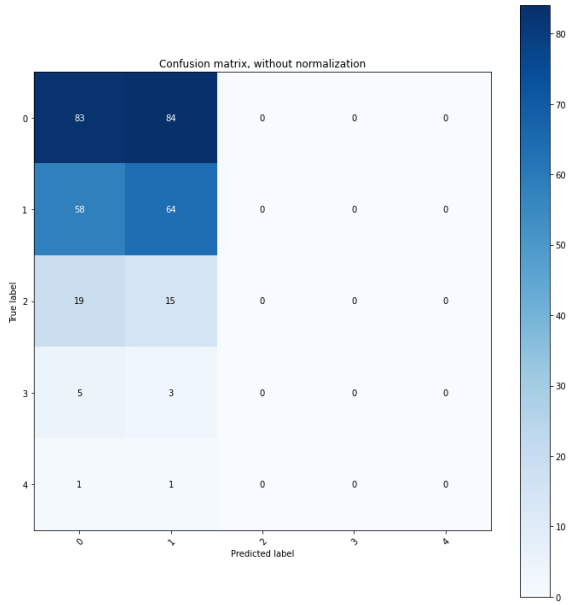

Figure 4. The confusion matrix of test

Then we proceeded to divide the dataset so as to be able to train the model and have a first test case, the proportion of test data was left on 1/3.

An Undersampling function was applied to the training sets as the dataset was unbalanced, in particular for the first two classes there was a sufficient number of samples, while for the other three the samples were not sufficient; therefore for class zero the samples were decreased, instead for classes one, two, three and four the samples were kept at their original number.

Several models were then tested: BernoulliNB, GaussianNB, RandomForesClasifier, Perceptron, SVM, GradientBoostingClassifier, modifying the various parameters for each model. The SVM model was shown to be the best for classification compared to the others. To further increase the classification capacity, the Grid Search algorithm was applied to SVM, which made it possible to find the most powerful classifier.

Through SVM an accuracy of approximately 55% has been reached, this is the model that worked in the best way among all those tested, the data have been normalized. Several parameters have been tested using the Grid Search for SVM, but in any case the level of accuracy remains just above 50%, this tells us that the model does not have a good classification capacity but it is the

best result that could be achieved without distorting the data, the further step would be to test the model on a new dataset.

3. Regression Task

Our aim in this task has been to train a model to predict the Minimum Closest Point of Approach (CPA), having the trajectories, speeds, angles, of 5 drones. So by formalizing the machine learning model we have had to create a model that approximated a function:

$$f : X -> Y \tag{3}$$

where $X \subseteq \Re^{35}$ and 35 is the number of features, $Y = \Re$ where an real rappresent CPA.

3.1. Normalize the dataset

For the normalization the MinMaxScaler method of the sklearn library was used, it rescales the dataset making all the features have a value between [0,1]. MinMaxScaler scales the values of a range without changing the shape of the original distribution. It is used by calculating: So first we split the dataset read through the **pandas** library into two parts: X which contains the 1000 rows with all features from 1 to 35; Y which contains the output associated with the 1000 rows of X. More precisely with the MinMaxScaler the formula for $x \in X$ is:

$$x\_std = \frac{x - x.min}{x.max - x.min} \tag{4}$$

$$x\_scaled = x\_std * (max - min) + min \tag{5}$$

The MinMaxScaler was used as it is more efficient when the data is not normally distributed, it is also less affected by outliers but narrows the data range of the outliers. Then it was necessary to split the dataset into training set and test set.

3.2. Split the Dataset

The dataset was split in this way: The training dataset had 975 samples, the test dataset 25 samples. The target distribution of entire dataset was represented from this histogram:

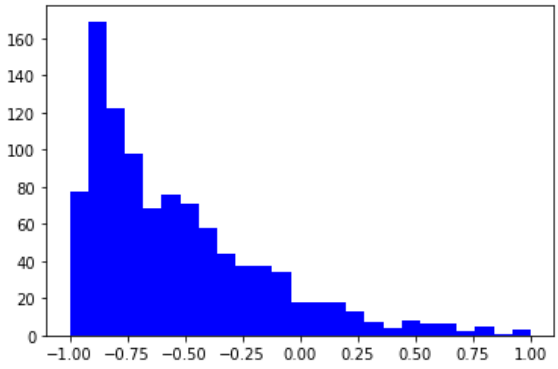

Figure 5. The distribution of target variables

3.3. The models and the setup

Three models were tested: Linear Regression, Linear Support Vector Regression (SVR), Polynomial SVR and Random Forest Regressor. We will analyze the test results later. But the best model for this problem was SVR polynomial regression. A polynomial regression model uses a form of regression analysis in wich the relationship beetween the indipendent variable x and the dependent variable y is modelled as an nth degree polynomial in x. The parameters that have been chosen for SVM polynomial regression are:

- Polynomial kernel;

- C with value 0.1;
- Degree with value 2;
- Scale gamma.

3.4. Tests

Various models and various configurations were used for the tests, the configurations and the regression score (R2) obtained as the first parameter for evaluating the model are shown below. In the next subsection, other metrics for performance evaluation will be analysed.

No particular configurations were chosen for the linear regression, the set parameters were left with their default value, in this case the model obtained an R2 value of -0.45, therefore very negative with respect to the model, which demonstrates that the model is not adequately suited to the data provided; later we tried to add the positive parameter which forces the coefficients to be positive, this further worsened the R2 bringing it to a value of -0.48. Hence this model was left. Two configurations were used for the SVM regression. The first SVM with a linear kernel

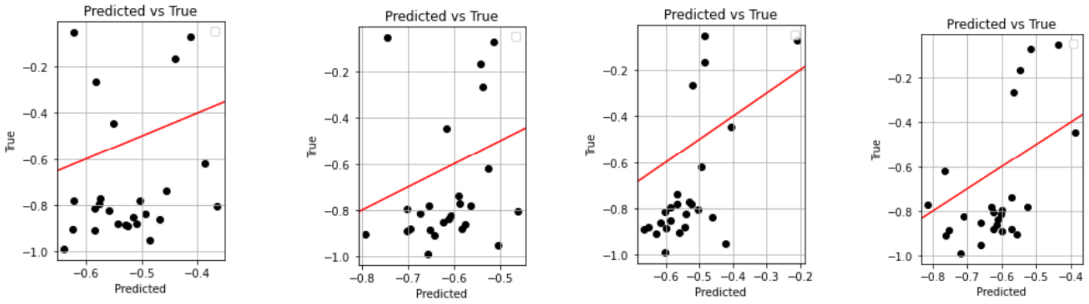

Figure 6. First image Predicted versus True for linear regression R2 = -0.45; second image Predicted versus True for linear SVR R2 = -0.0.3; third image Predicted versus True for random forest regressor R2 = -0.10; fourth image Predicted versus True of polynomial regression R2 = -0.10

and a value for the parameter C=1.5 for a value of R2 and -0.08, this model also does not fit the data well; even the configuration with C=0.01 did not improve the R2 which dropped to -0.03, a non-optimal result, trying to further lower the value of C the R2 did not improve.

The second configuration tested was the one with the polynomial kernel; in this case different values for the parameter C and the degree were tried, starting from C = 5 and degree = 4 and obtaining an R2 of -0.17, and then gradually lowering them until reaching C=0.1 and degree = 2 to get an R2 of 0.16; then we tried to lower the parameter C again, bringing it to 0.001 with a degree = 2, the result was an R2 of -0.09, therefore the chosen configuration was the one that gave an R2 of 0.16. For the random forest model, the first test was performed leaving the default values, this led to an R2 of -0.09; from there we tried to raise the number of estimators from 100 to 1000 obtaining an R2 of -0.08.

In the graphs of figure 6 we can see the prediction capability of the model, ideally the black points which are the true values should be close to the regression line, where Predicted = True.

3.5. Performance evaluation

The tests were carried out by training the four models considered in the section above, the parameters with which the models were evaluated are: Mean squared error (MSE), R-squared (R2) and Mean Absolute Error (MAE). Mean Squared Error, measures the average of the squares of the errors — that is, the average squared difference between the estimated values and the actual value. We used R-squared because it indicates the portion of the variance in the dependent variable that is predictable from the independent variable. MAE it is simply the average absolute vertical

or horizontal distance between each point in a scatter plot and the Y=X line. In other words, MAE is the average absolute difference between X and Y. Furthermore, each error contributes to MAE in proportion to the absolute value of the error.The results obtained from the tests are:

- For linear regression, MSE was 0.11 with an R2 of -0.45 and a MAE of 0.31;
- For linear SVR, MSE was 0.08 with an R2 of -0.03 and a MAE of 0.25;
- For polynomial SVR, MSE was 0.06 with an R2 of 0.16 and a a MAE of 0.23;
- For the random forest, MSE was 0.08 with an R2 of -0.08 and a MAE of 0.27.

We can see the prediction of the model, in the graph Predicted vs true, the line created by the model follows the direction of the data, but the data stays away from the line, this leads to see that the MSE actually has a low value the difference for the points between the forecast and the real value is not very wide, the R2 tells us that the line is positive.

For the regression task, we first normalized the dataset because the data were expressed in different scales as for the classification, this allowed us to increase the training target of the model. The dataset was then subdivided by testing some proportions, in particular that of 975 samples for training and 25 for testing was chosen.

Different models were tested for the test: Linear Regression, Linear Support Vector Regression (SVR), Poly- nominal SVR and Random Forest Regressor. As for the classification, the SVM regressor performed better than the other models. The tested models were evaluated through the values of the Mean Squared Error (MSE), the R squared and the Mean Absolute Error (MAE).

The best result is offered by the SVM regressor model which has a very low error, even if the R2 tells us that the model explains 0.16% of the variance of the target variable, which is very low value. On average, forecasts have a squared error of 0.08% (MSE value), remembering that one of the main goals of learning is to minimize the error on the dataset. Tests were performed by changing model parameters, for example by modifying the degree of the polynomial, but the best result remains the highlighted one.

References

[1] https://towardsdatascience.com/machine-learning-with-python-classification-complete-tutorial-d2c99dc524ec

[2] https://towardsdatascience.com/machine-learning-with-python-classification-complete-tutorial-d2c99dc524ec

[3] https://towardsdatascience.com/machine-learning-with-python-classification-complete-tutorial-d2c99dc524ec

[4] https://towardsdatascience.com/machine-learning-with-python-regression-complete-tutorial-47268e546cea

[5] https://towardsdatascience.com/ways-to-evaluate-regression-models-77a3ff45ba70