SAPIENZA
UNIVERSITÀ DI ROMA

Machine Learning a.y. 22-23

Homework 1: Report

Davide Gabrielli - 1883616
gabrielli.1883616@studenti.uniroma1.it

December 9, 2022

1 Introduction

In this report I will describe the implementation for the given two tasks (classification and regression) about collision detection of drones in a 2D environment.

1.1 Dataset

The dataset is the one provided by the course, it contains 1000 samples of drone collisions in a 2D environment. The dataset contains the following features:

- **For 5 UAVs** (where i is the UAV number):
  - **UAV_i_track**: clockwise angle from north between the i-th UAV and its target (0, 2*pi)
  - **UAV_i_x**: x coordinate of the i-th UAV
  - **UAV_i_y**: y coordinate of the i-th UAV
  - **UAV_i_vx**: x velocity of the i-th UAV
  - **UAV_i_vy**: y velocity of the i-th UAV
  - **UAV_i_target_x**: x coordinate of the i-th UAV target
  - **UAV_i_target_y**: y coordinate of the i-th UAV target
- **num_collisions**: number of collisions for the sample (0, 1, 2, 3, 4)
- **min_CPA**: minimum CPA in the sample (an estimated point in which the distance between two objects, of which at least one is in motion, will reach its minimum value)

The dataset is unbalanced, as it contains 1000 samples divided into 5 classes that represent the number of collisions in the sample, as shown in (Fig. 1). This is also due to the fact that the distribution of the number of collisions is not uniform.

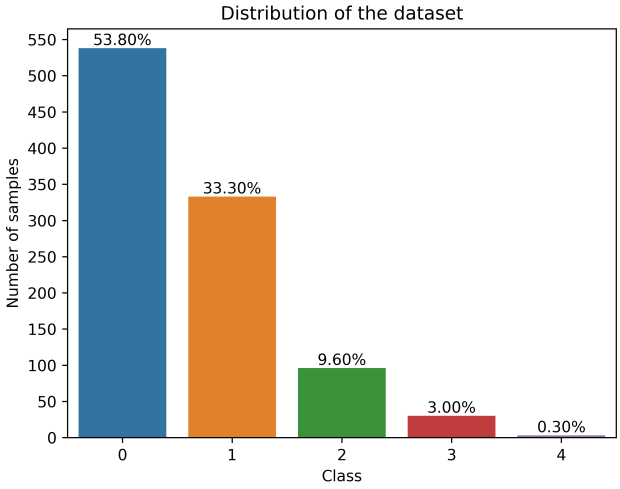

Figure 1: The dataset is composed of 1000 samples divided into 5 classes that represent the number of collisions.

1.2 Classification

The classification task is to predict the number of collisions in a sample. I will use several classification algorithms to compare their performance. The algorithms I will use are:

- **Logistic Regression**
- **Random Forest**
- **Support Vector Machine**
- **Gaussian Naive Bayes**

Since the dataset is unbalanced, I expect better results from Random Forest and Support Vector Machine, as they are able to handle unbalanced datasets. The following metrics will be used to evaluate the performance of the algorithms:

- **Accuracy**: measures how often the classifier makes the correct prediction
- **Precision**: measures the model's ability to avoid false positives
- **Recall**: measures the model's ability to avoid false negatives
- **F1-Score**: the harmonic mean of precision and recall

Since the dataset is unbalanced, I will use the F1-Score as the main metric to evaluate the performance of the algorithms.

1.3 Regression

The regression task is to predict the minimum CPA in a sample. I will use the following regression algorithms to compare their performance:

- **Support Vector Regression**
- **Gradient Boosting Regression**

The following metrics will be used to evaluate the performance of the algorithms:

- **Mean Squared Error**: the average of the squared differences between the predictions and the actual values
- **R2 Score**: the coefficient of determination, which is a measure of how well the predictions fit the actual values

2 Classification Task

In this section I will describe how I managed the classification task, the way the dataset has been preprocessed and the results obtained with the different models.

2.1 Data Preprocessing

2.1.1 Feature selection

The first step is to select the features that will be used for the classification task. The dataset for each drone contains the following 7 features: the position, the velocity and the target position of the drone and the angle between the drone and the target (relative to the north).

I have decided to remove the angle from the dataset because of the redundancy, since it can be computed from the velocity or the target position. So now the dataset is composed by $6 \times 5 = 30$ features for the 5 drones in the environment plus the label, which is the number of collisions.

2.1.2 Normalization

Since the dataset contains features with different ranges, it is necessary to normalize the dataset. My approach is to normalize each row separately. In fact we can imagine each row as an "environment" with 5 drones referring their coordinates to the point (0,0) in space. So we can normalize in a way that the "environment" keeps the same shape, but the values are in the range [0,1]. To achieve this we have to distinguish two types of values: absolute and relative.

The position of the drone is an absolute value (refer to the point (0,0) in space), while the velocity is a relative value (it's the difference between the current position and the previous one) so it's not good to use the same normalization for both of them. So this is the approach that I have decided to use:

- **Positions**: since position and target position have an absolute meaning, I have used the min-max normalization for each row with the following formula:

$$\text{norm}(x_{D^i}) = \frac{x_{D^i} - minCoord}{maxCoord - minCoord} \qquad \text{norm}(y_{D^i}) = \frac{y_{D^i} - minCoord}{maxCoord - minCoord} \tag{1}$$

  Where:

$$\begin{aligned} maxX &= \max_{1 \le i \le 5} x_{D^i} \\ maxY &= \max_{1 \le i \le 5} y_{D^i} \\ minX &= \min_{1 \le i \le 5} x_{D^i} \\ minY &= \min_{1 \le i \le 5} y_{D^i} \\ maxCoord &= \max(maxX, maxY) \\ minCoord &= \min(minX, minY) \end{aligned}$$

In this way, after normalization, the position of the drones (and their target positions) will be in the range [0, 1].

The sample now represents a new "environment" where the bottom-left corner corresponds at the position of the drone (or the target position) with the smallest x and y coordinates and it will also preserve the aspect ratio of the original one. The choice of preserving the aspect ratio is due to testings showing better results.

- **Velocity**: since it has a relative meaning, have been used the predefined $maxCoord$ and $minCoord$ values to normalize the data with the following formula:

$$\text{norm}(vx_{D^i}) = \frac{x_{D^i}}{maxCoord - minCoord} \qquad \text{norm}(vy_{D^i}) = \frac{y_{D^i}}{maxCoord - minCoord} \quad (2)$$

2.1.3 Splitting

The dataset has been splitted in training and test set with a ratio of 80/20 in a stratified way, so that the distribution of the classes is the same in the training and test set.

2.1.4 Balancing

The dataset is unbalanced, so it is necessary to balance it. I have tried different methods for oversampling but since there are < 3 samples of the minority class after splitting it, the standard oversampling (such as SMOTE or Random Over Sampling) methods are not really effective. Therefore, I have decided to implement my own oversampling method.

The idea is to use the semantic meaning of the dataset to generate new samples. Using the same concept of the normalization, we can imagine each sample as an "environment" with 5 drones. So I have decided to generate new samples by changing the position of the drones, keeping the velocity and the angle between the drone and the target constant. In this way it is possible to move the drones backwards (using the inverse of velocity) to generate new samples having the same number of collisions.

Since the probability of the event is not uniform (e.g. the probability of having 4 collisins is lower than the probability of having 3 collisions or 2 collisions, etc.), it makes sense to keep a similar distribution of the classes after oversampling. So I have decided to generate new samples generated for each class (and not just for the minority class) in a way that the distribution of the classes keeps its trend. In details, the number of new samples generated for each class is as in Table 1.

	Class 0	**Class 1**	**Class 2**	**Class 3**	**Class 4**
Samples generated	50	55	65	70	75

Table 1: Number of samples for each class

As we can see, there are more samples generated for the classes with less samples but in a way that the distribution of the classes remains similar to the original one.

2.2 Training

2.2.1 Classification models

I have compared different classifiers for this task.

- **Logistic Regression**: LogisticRegression from the scikit-learn library.
- **Random Forest**: RandomForestClassifier from the scikit-learn library.
- **Support Vector Machine**: SVC from the scikit-learn library.
- **Gaussian Naive Bayes**: GaussianNB from the scikit-learn library.

2.2.2 Hyperparameter tuning

I have used the GridSearchCV method to tune the hyperparameters of the classifiers, using the F1 score as the metric to optimize. The hyperparameters that I have tuned are:

- **Random Forest**: n_estimators, max_features, criterion
- **Support Vector Machine**: C, gamma, kernel
- **Logistic Regression**: C, penalty, solver
- **Gaussian Naive Bayes**: var_smoothing

2.3 Evaluation

The evaluation of the models is performed using the F1 score, the accuracy and the confusion matrix over the test set, testing on 30 seeds. Since the dataset is unbalanced it is better to use the F1 score instead of the accuracy. As shown in Figure 2, the Support Vector Machine classifier has the best F1 score, followed by the Logistic Regression classifier.

For further analysis, the confusion matrix of the 4 classifiers is shown in Figure 3 and the accuracy is shown in Table 2. Also a plot with the prediction against the ground truth is shown in Figure 5 to have a better visualization of the performance of the classifiers and to check if they also predict the minority class.

We summarize the results in Table 2.

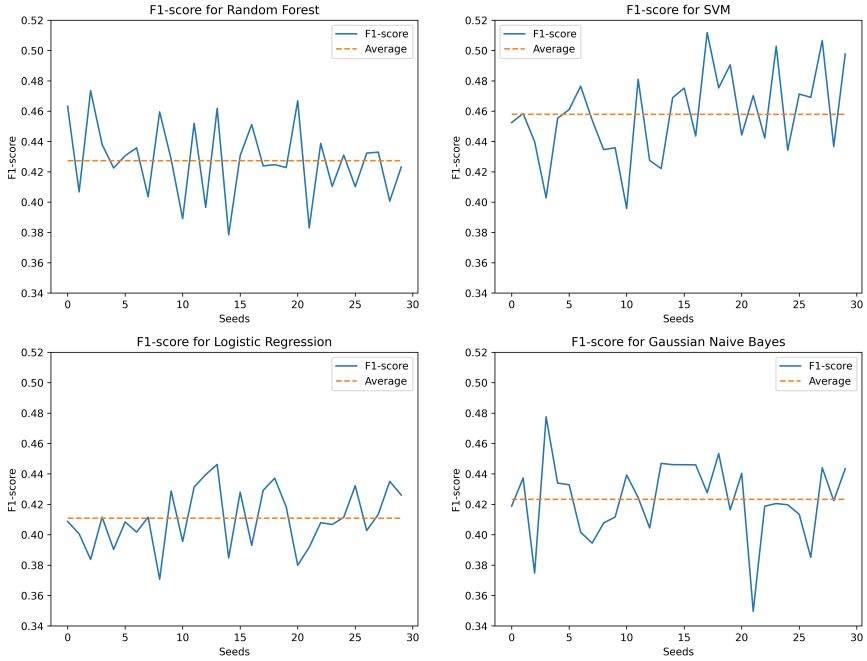

Figure 2: F1 score of the different classifiers on the test set

Model	Accuracy	F1 Score
Random Forest	0.51116	0.42739
Support Vector Machine	0.47850	0.45794
Logistic Regression	0.49133	0.41088
Gaussian Naive Bayes	0.44467	0.42329

Table 2: Evaluation of the regression task

Figure 3: Confusion matrix of the different classifiers on the test set

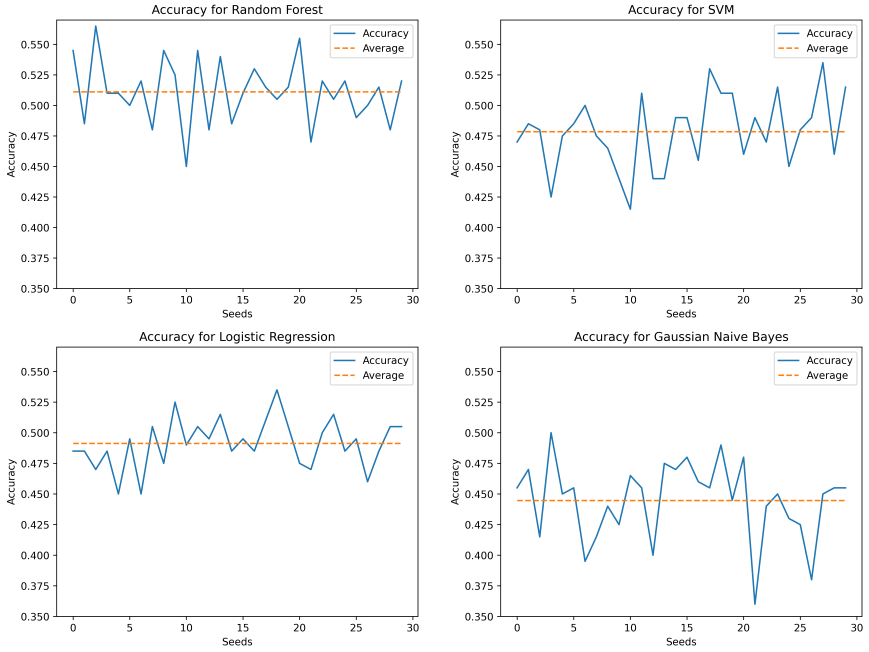

Figure 4: Accuracy of the different classifiers on the test set

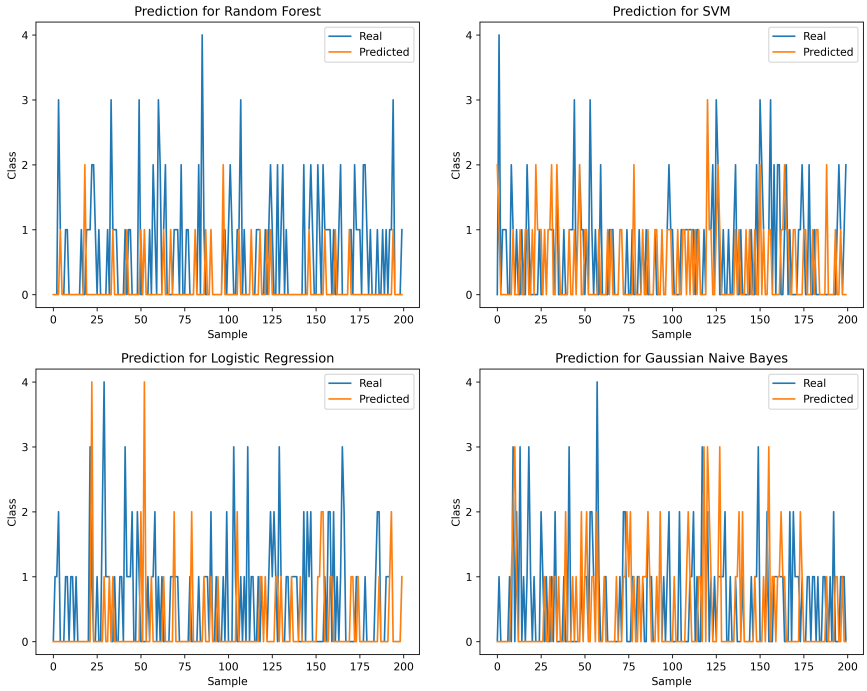

Figure 5: Prediction of the different classifiers on the test set

3 Regression Task

In this section I will describe how I managed the regression task, the way the dataset has been preprocessed and the results obtained with the different models.

3.1 Data Preprocessing

3.1.1 Feature selection

The first step is to select the features that will be used for the regression task. The dataset for each drone contains the following 7 features: the position, the velocity and the target position of the drone and the angle between the drone and the target (relative to the north).

Also in this case, I have removed the angle from the dataset since it can be computed from the positions and the velocity. The dataset is then composed by $6 \times 5 = 30$ features for the 5 drones in the environment and the label, which is the minimum CPA (Closest Point of Approach) between the drones.

3.1.2 Normalization

Since the dataset contains features with different ranges, it is necessary to normalize the dataset. Also in this case, the normalization is performed for each row separately respecting the same approach described in Section 2.1.2.

In this task, also the label have to be normalized and the normalization is performed using the min-max normalization, which is defined as:

$$\text{norm}(minCPA) = \frac{minCPA - \min(minCPA)}{\max(minCPA) - \min(minCPA)} \tag{3}$$

Where:

$min(minCPA)$ = is the minimum value of minCPA in the dataset
$max(minCPA)$ = is the maximum value of minCPA in the dataset

3.1.3 Splitting

The dataset has been splitted in training and test set with a ratio of 80/20.

3.2 Training

3.2.1 Regression models

I have implemented the following regression models:

Model	R2 score	MSE
Support Vector Regression	-0.09102	0.03922
Gradient Boosting Regression	-0.00721	0.03639

Table 3: Evaluation of the regression task

- **Support Vector Regression**: SVR model from the scikit-learn library.
- **Gradient Boosting Regression**: GradientBoostingRegressor model from the scikit-learn library.

3.2.2 Hyperparameter tuning

I have used the GridSearchCV method to tune the hyperparameters of the classifiers, using the R2 score as the metric to optimize. The hyperparameters that I have tuned are:

- **Support Vector Regression**: C, gamma, kernel
- **Gradient Boosting Regression**: n_estimators, max_features, learning_rate, loss

3.3 Evaluation

The evaluation of the regression task is performed using the R2 score and the mean squared error, shown in Table 3.

Also in this case, a plot of the prediction of the minCPA against the ground truth using the different regression models is shown in Figure 6.

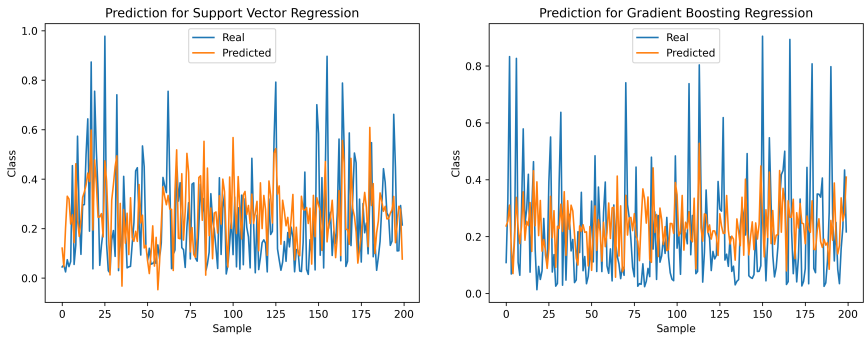

Figure 6: Prediction of the minCPA for the test set using the different regression models

4 Conclusions

The results obtained for both the classification and the regression task are not satisfactory and the models are not able to predict the collision risk with a good accuracy.

For the classification task, all the models have reached a F1 score lower than 0.5. The best F1 score has been obtained by the Support Vector Machine classifier (0.4579).

For the regression task, the Gradient Boosting Regression has the best R2 score (-0.00721) and the Support Vector Regression has the best MSE (0.03922).

Since the bad performances achieved by all the algorithms, it is not possible to conclude which one is the best for this dataset. The probably reasons for the poor performance of the models are the complexity of the problem, which has a lot of features, and the unbalanced dataset, which has a lot of samples of the majority class and few samples of the minority class.