SAPIENZA
UNIVERSITÀ DI ROMA

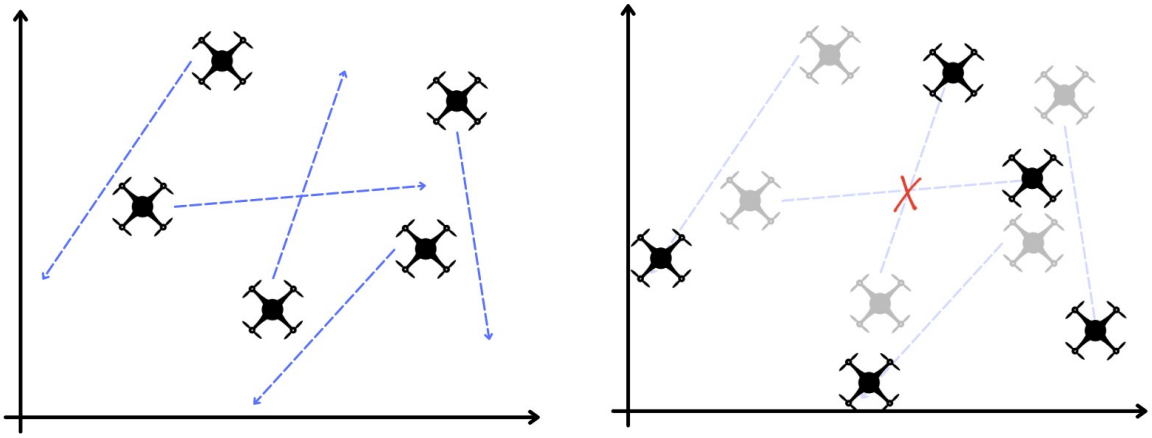
Machine Learning a.y. 22-23

Homework 1: Report

Franesco Palandra
1849712

December 30, 2022

1 Introduction



(a) packet delivery ratio

(b) packet mean delivery time

Figure 1

In this problem, there are 5 drones, for each one we know his clockwise angle from the north between the UAV and its target, the position x and y, the position of the target destination tx and ty, and lastly the speed components along the axis vx and vy.

2 Classification

For the classification part, knowing the initial condition the number of collisions has to be established.

2.1 Data Preprocessing

In the early stage, before building a model, we need to preprocess the data. As we can see from the image below 2, the number of the first two classes is higher than the last two, with just a few data in the dataset.

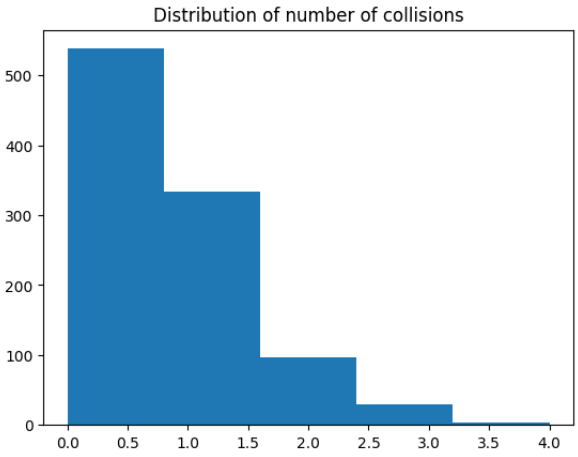
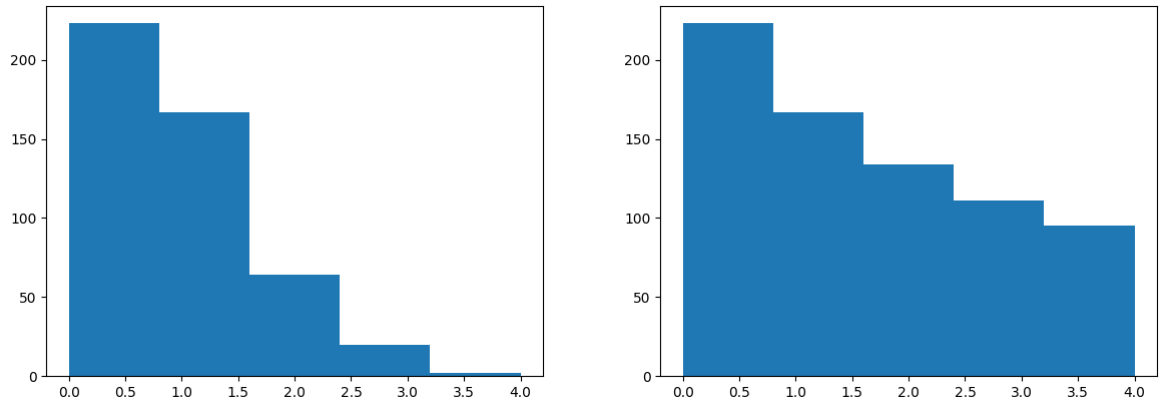


Figure 2: Number of collisions

First of all, all the nan values have to be handled, but since in the dataset, there are no such data this step can be skipped.

2.1.1 Undersampling and Oversampling

Since the dataset is unbalanced, the bias in the training dataset can influence the model, leading some to ignore the minority class entirely. One approach to addressing the problem of class imbalance is to randomly resample the training dataset. The two main approaches to randomly resampling an imbalanced dataset are to delete examples from the majority class, called undersampling, and to duplicate examples from the minority class, called oversampling. Resampling involves creating a new transformed version of the training dataset in which the selected examples have a different class distribution.



(a) classes distribution after the undersampling

(b) classes distribution after the oversampling

Figure 3

2.1.2 Normalization of the data

The data then is normalized with the min-max normalization. This method preserves the relationships among the original data values and gets all the scaled data in the range (0, 1) following the equation below:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

Normalization is essential as variables that are measured at different scales do not contribute equally to the model fitting and model learned function and might end up creating a bias. Thus, to deal with this potential problem feature-wise normalization such as MinMax Scaling is usually used prior to model fitting.

2.1.3 Feature Importance Analysis

Feature importance indicates how much each feature contributes to the model prediction. Basically, it determines the degree of usefulness of a specific variable for a current model and prediction. By analyzing variable importance scores, it is possible to find out irrelevant features and exclude them. Reducing the number of not meaningful variables in the model may speed up the model or even improve its performance.

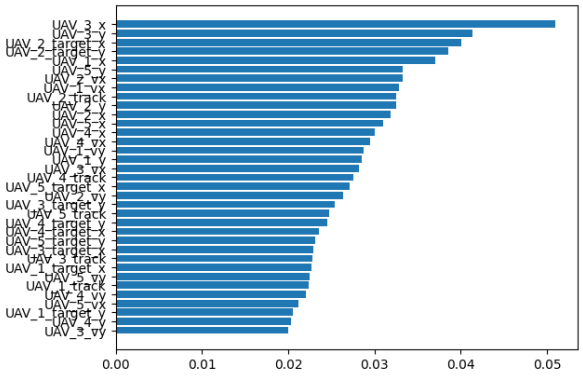


Figure 4: Number of collisions

On the entire dataset, a random forest was used to recognize the most relevant features. This enabled the classification of the importance of the characteristics. In the end, no feature has been removed and each one is considered to be relevant for the fitting.

2.1.4 Outliers

Outliers are those data points that differ significantly from other observations present in the given dataset. It is risky to include outliers in data driven models. The existence of one single misleading value has the potential to change the conclusion implied by the model. Is is therefore, important to detect and then decide whether to remove it or not from the dataset. Sometimes the data point may be extremely high or low but that does not mean it is an outlier that we want to get rid of. It may be simply an extreme data point. Z-score is used to convert the data into another dataset with mean = 0.

$$Z = \frac{x_i - \bar{x}}{\sigma} \tag{2}$$

2.2 Models

The models used for this problem are:

- Decision Tree
- Random Forest
- SVC
- KNN

Each model has been fitted before and after the data preprocessing and there is an improvement in the f1_score.

2.2.1 Decision Tree Classifier

	precision	recall	f1-score	support
0.0	0.53	0.45	0.49	177
1.0	0.29	0.32	0.30	110
2.0	0.13	0.16	0.14	32
3.0	0.00	0.00	0.00	10
4.0	0.00	0.00	0.00	1
accuracy			0.36	330
macro avg	0.19	0.19	0.19	330
weighted avg	0.40	0.36	0.38	330

Figure 5: Classification Report of DT

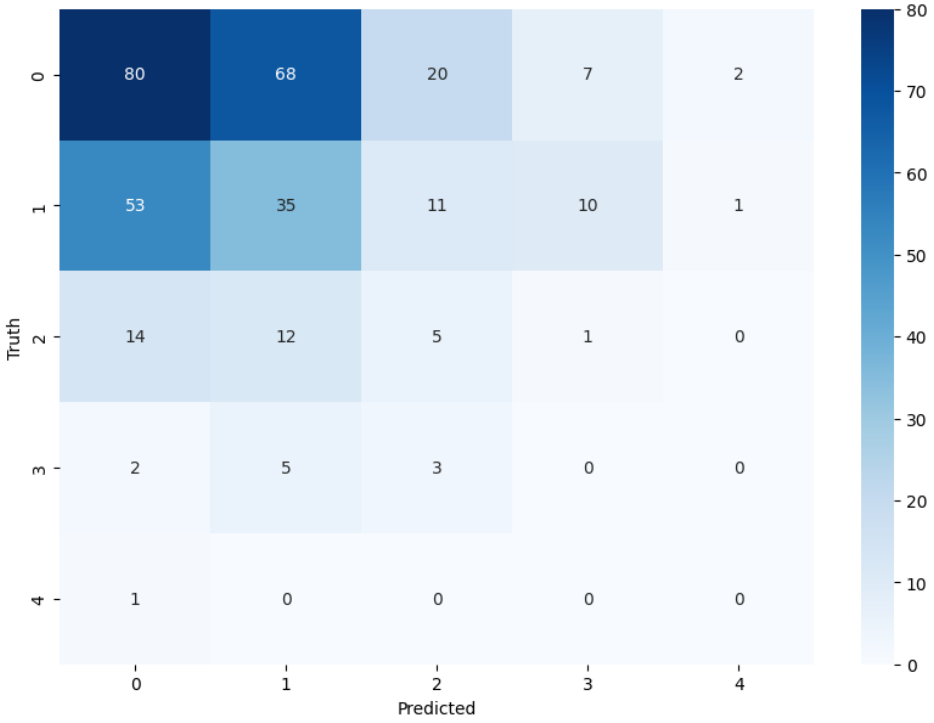


Figure 6: Confusion table

2.2.2 Random Forest

Random forest creates decision trees on randomly selected data samples, gets predictions from each tree and selects the best solution by means of voting.

	precision	recall	f1-score	support
0.0	0.57	0.81	0.67	177
1.0	0.40	0.25	0.30	110
2.0	0.09	0.03	0.05	32
3.0	0.00	0.00	0.00	10
4.0	0.00	0.00	0.00	1
accuracy			0.52	330
macro avg	0.21	0.22	0.20	330
weighted avg	0.45	0.52	0.46	330

Figure 7: Classification Report of RF

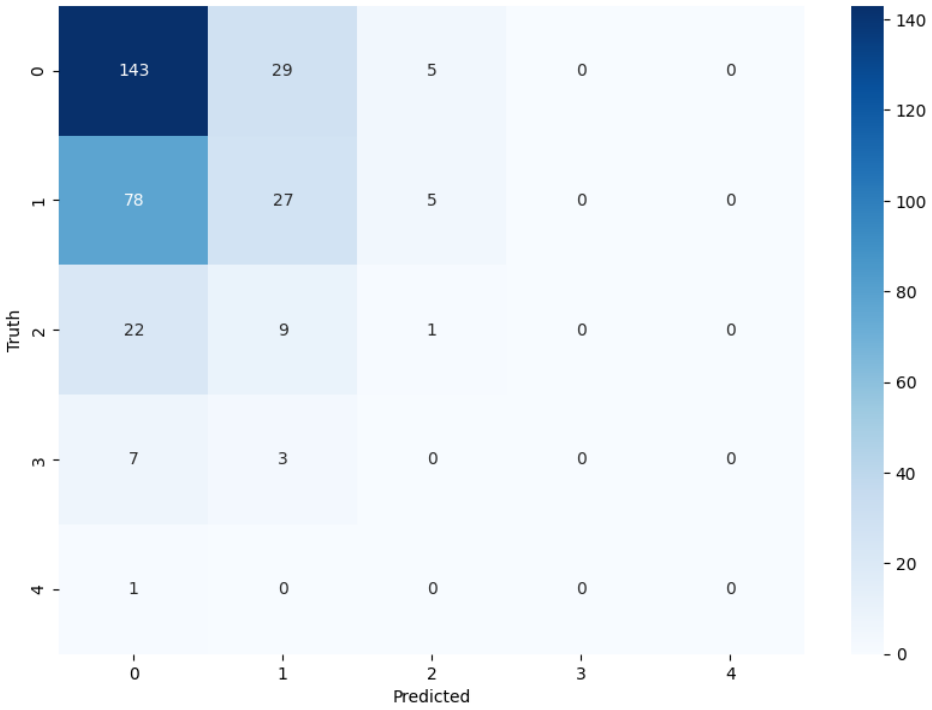


Figure 8: Confusion table

2.2.3 SVC

Support Vectors Classifier tries to find the best hyperplane to separate the different classes by maximizing the distance between sample points and the hyperplane.

	precision	recall	f1-score	support
0.0	0.56	0.84	0.67	177
1.0	0.38	0.17	0.24	110
2.0	0.22	0.06	0.10	32
3.0	0.25	0.10	0.14	10
4.0	0.00	0.00	0.00	1
accuracy			0.52	330
macro avg	0.28	0.24	0.23	330
weighted avg	0.46	0.52	0.45	330

Figure 9: Classification Report of SVM

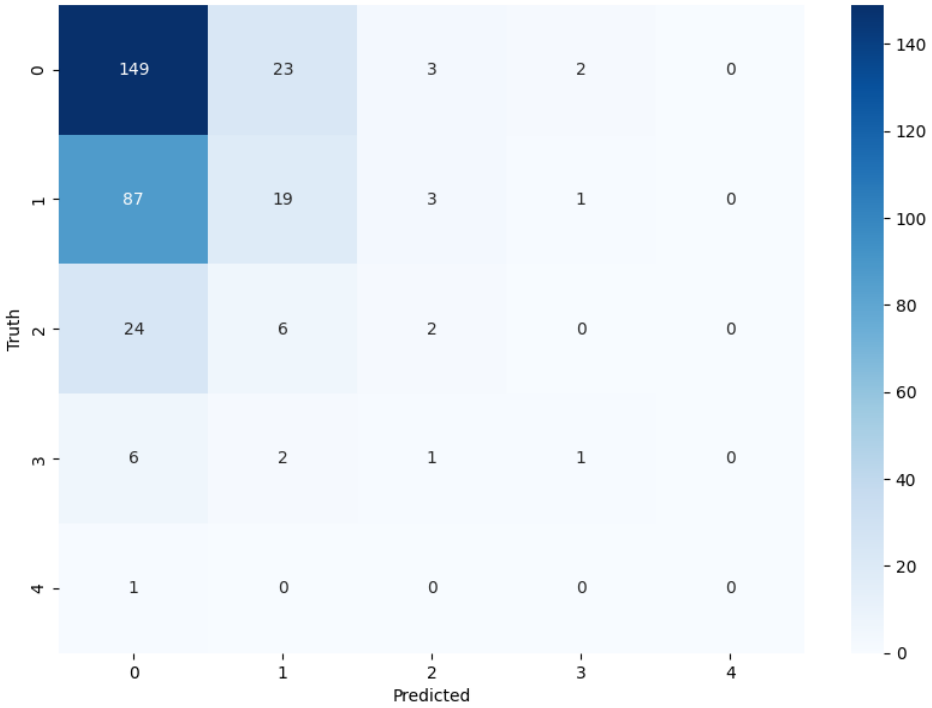


Figure 10: Confusion table

2.2.4 KNN

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. It works off the assumption that similar points can be found near one another.

	precision	recall	f1-score	support
0.0	0.55	0.34	0.42	177
1.0	0.35	0.27	0.31	110
2.0	0.12	0.31	0.18	32
3.0	0.04	0.20	0.07	10
4.0	0.00	0.00	0.00	1
accuracy			0.31	330
macro avg	0.21	0.22	0.19	330
weighted avg	0.42	0.31	0.35	330

Figure 11: Classification Report of KNN

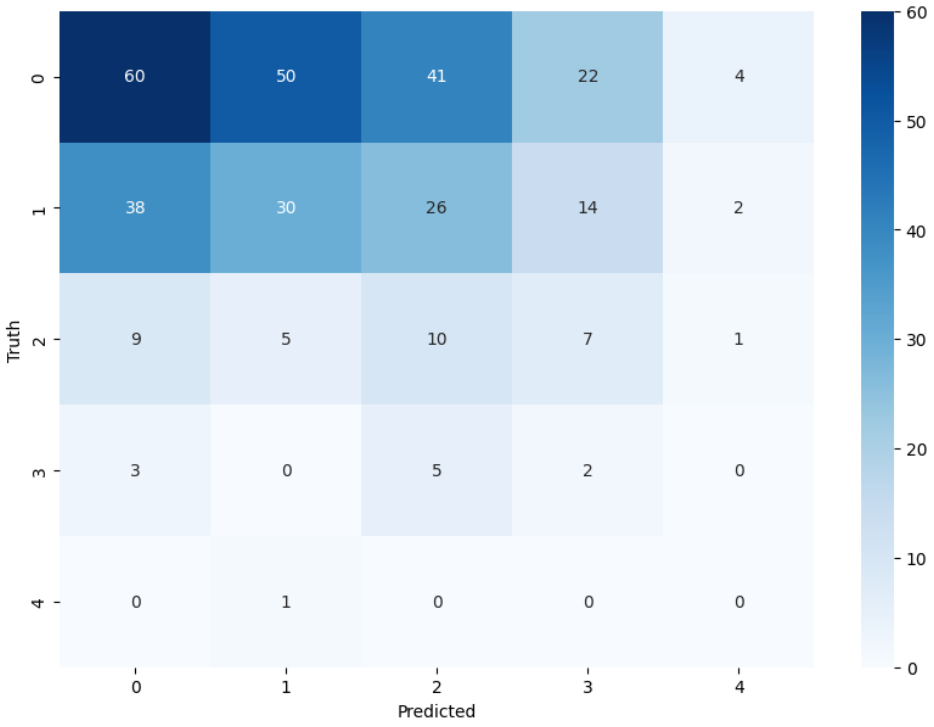


Figure 12: Confusion table

2.2.5 Model evaluation

In order to evaluate the performance of each model, it has been used the cross-validation. The training set is split into k smaller sets:

- A model is trained using the folds as training data;
- The resulting model is validated on the remaining part of the data

The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop.

2.2.6 Results

For the results for each model have been done in different runs with different parameters. In order to analyze the effect of the preprocessing the results between the models with and without these changes have been compared. Although the accuracy remains similar, the value of the f1-score increase with preprocessing. Below is the table with the final results of each model after data preprocessing.

name	accuracy	f1-macro	cv-accuracy	cv-f1-macro
DT	0.36	0.19	0.4220 (+/- 0.0136)	0.2064 (+/- 0.0389)
RF	0.52	0.20	0.5130 (+/- 0.0383)	0.1866 (+/- 0.0475)
SVC	0.52	0.23	0.5260 (+/- 0.0214)	0.1745 (+/- 0.0490)
KNN	0.31	0.19	0.4820 (+/- 0.0680)	0.2151 (+/- 0.0528)

To choose the best model I considered these values in order: cv_accuracy, cv_f1macro, f1_macro, accuracy. I selected SVC as the best model and I proceed with the hyper-parameter tuning.

2.2.7 Model Tuning

The purpose of tuning a model is to ensure that it performs at its best. This process involves adjusting various elements of the model to achieve optimal results. By fine-tuning the model, it is possible to get the highest rate of performance possible. For the tuning I used the grid search, a set of values for each hyperparameter has defined to form a grid. Different combinations of these hyperparameter values are tried and the combination which yields the best result is selected as the final set of optimal hyperparameters. For the hyperparameters the choice are:

- C: [0.5, 1]
- kernel: ['rbf', 'sigmoid']
- gamma : ['scale', 'auto']
- 'decision_function_shape': ['ovo', 'ovr'],
- random_states

2.2.8 Conclusion

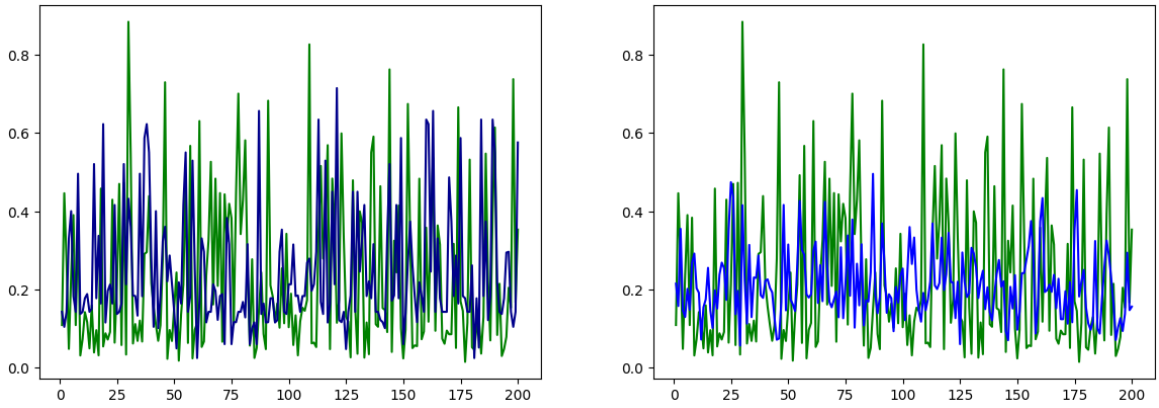
The outliers were removed from the entire dataset using zscore, assuming there was noise in the dataset. The dataset has been divided into two sections: train (66%) and test (33%). Five distinct models were trained, tested, and evaluated. It was discovered that the unbalanced dataset resulted in excellent accuracy but a very poor f1_macro. The larger classes were undersampled, while the smaller classes were oversampled. The new class samples distribution is: c0 = 1/3, c1 = 1/4, c2 = 1/5, c3 = 1/6, c4 = 1/7. This division still keeps the classes unbalanced, but it is much less severe than it was at the start. Furthermore, I think that this sample division is more useful for solving the problem at hand than having the same number of samples for all classes. This is because, given the nature of the problem (prediction of the number of collisions), I think that the first classes will be far more populated than the last classes. With the new dataset, models were trained, tested, and evaluated. These demonstrate a significant improvement in f1_macro. SVC was chosen as the best model and it was tuned. Because of the computational complexity of SVC, the tuning was done on a smaller grid than I would have preferred.
The SVM with the following parameters $\{'C' : 1,' decision\_function\_shape' :' ovo',' gamma' :' scale',' kernel' :' rbf',' random\_state' : 0\}$ is the best model for solving the problem.

3 Regression

For the regression part, the main goal is to predict the minimum Closest Point of Approach (CPA) among all the possible pairs of UAVs. The CPA is the minimum distance between drones over time. Knowing the initial positions of the target and the speed the CPA could be calculated analytically with vectors, deriving over time and taking the minimum. However, this is not the purpose of the exercises. Four different models for regression have been studied with different hyperparameters and below are the results (Linear regression was not studied due to the bad results).

- Decision Tree with {max_depth=10, min_samples_leaf=5}
- Random Forest with {n_estimators=200, max_depth=10}
- KNN with {n_neighbors=5, weights='distance'}
- SVR with {kernel = 'poly', gamma=0.3, degree=2, C=0.3}

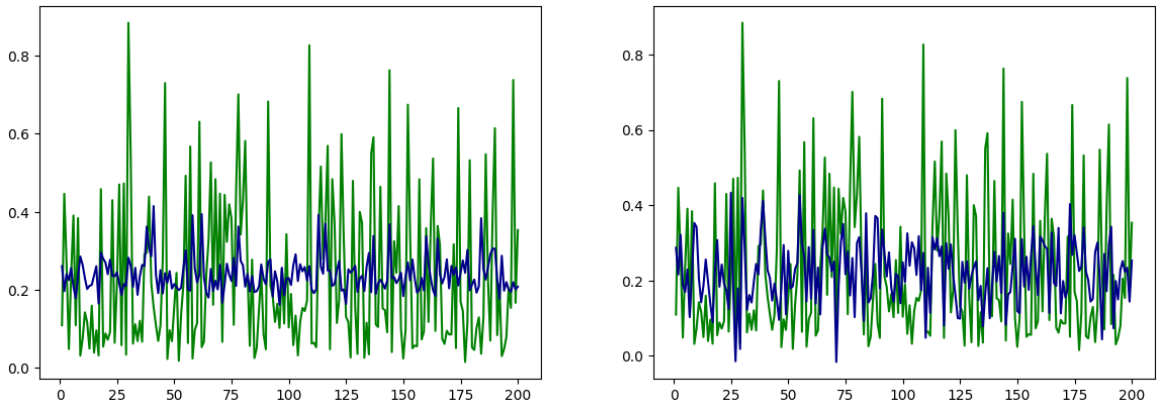
In green is the actual CPA while in blue is the CPA calculated by the different models



(a) Linear Regression

(b) Decision Tree

Figure 13



(a) Random Forest

(b) SVR

Figure 14

	DT	KNN	RF	SVR
r2 score	-0.41	-0.04	0.03	0.11

As we can see from the images and the table above the first two methods LR and DT fit the data very poorly, while the other two work slightly better with SVR as the best model among the four studied.

4 Conclusions

In conclusion, for both classification and regression SVM turned out to be the best model. However, the results are not too good. In the classification, in particular, the dataset was unbalanced and some preprocess was needed in order to generate more data on the minority classes. That helped, but not too much, yet just few cases in class 3 or 4 got detected. In the regression, instead, Decision Trees and KNN seem to not fit well, SVR, however, had the best result with a r2-score of 0.11, not a perfect score but still decent.