Lorenzo Cirillo 1895955

REPORT HOMEWORK 1

CLASSIFICATION PROBLEM

We had to deal with an imbalanced dataset related to uavs.

To import the dataset, I used pandas as suggested by the tutor. By doing this, I could access the columns of the dataset in a simple way.

Then I prepared the dataset to be split. By dropping the columns 'min_CPA' and 'num_collisions', I obtained the features for the classification. By holding only the column 'num_collisions', I obtained a frame with the things we want to predict.

At this point I proceeded with the normalization of the dataset with values between 0 and 1

	UAV_1_track	UAV_1_x	UAV_1_y	...	UAV_5_vy	UAV_5_target_x	UAV_5_target_y
0	0.003575	0.167803	0.161493	...	0.482930	0.708205	0.694046
1	0.640368	0.422673	0.741763	...	0.337113	0.488949	0.271890
2	0.292795	0.407523	0.804773	...	0.786650	0.895128	0.553126
3	0.576375	0.245460	0.418722	...	0.220912	0.196630	0.589351
4	0.368716	0.817790	0.225677	...	0.138896	0.817285	0.611065
..	...	...	...	...	...	...	...
995	0.169678	0.123388	0.543553	...	0.670426	0.925073	0.571382
996	0.472615	0.320896	0.894419	...	0.725237	0.782904	0.803012
997	0.129801	0.402998	0.684298	...	0.122919	0.315931	0.135323
998	0.699582	0.794742	0.308183	...	0.280094	0.306320	0.522590
999	0.232669	0.404775	0.105787	...	0.969388	0.244723	0.835880

[1000 rows x 35 columns]

A dataset is imbalanced when there is an important difference between the number of samples of some classes.

I knew that the dataset given is imbalanced, but initially I tried to work without any kind of sampling on it.

The normalized dataset has been split in a training portion (2/3 of the data) and a test portion (1/3 of the data).

I set the first model which is svm and the second one which is random forest.

The metrics used for evaluating these models for our classification problem are accuracy, precision, recall, f1-score, confusion matrix, k-cross validation and roc curve.

We know that:

- the accuracy is simply a ratio of correctly predicted observations to the total observations.

- the precision is the ratio of the correctly predicted positive observations to the total predicted positive observations.

- the recall is the ratio of correctly predicted positive observations to all observations in actual class.

- f1-score is the weighted average of precision and recall.

By training and testing the models, the results are the following ones:

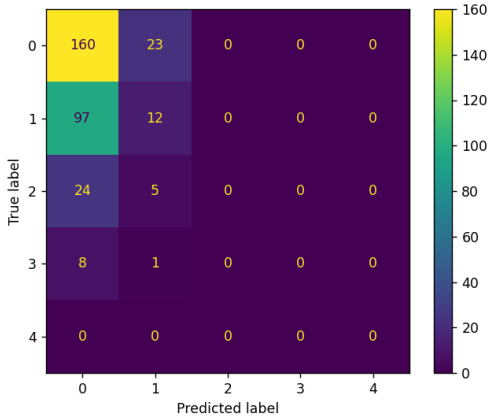
SVM table

	precision	recall	f1-score	support
0	0.568	0.957	0.713	184
1	0.350	0.072	0.120	97
2	0.000	0.000	0.000	38
3	0.000	0.000	0.000	9
4	0.000	0.000	0.000	2
accuracy			0.555	330
macro avg	0.184	0.206	0.166	330
weighted avg	0.419	0.555	0.432	330

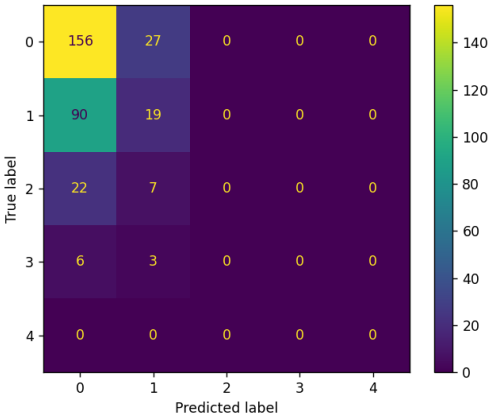
RF table

	precision	recall	f1-score	support
0	0.561	0.870	0.682	184
1	0.289	0.134	0.183	97
2	0.000	0.000	0.000	38
3	0.000	0.000	0.000	9
4	0.000	0.000	0.000	2
accuracy			0.524	330
macro avg	0.170	0.201	0.173	330
weighted avg	0.398	0.524	0.434	330

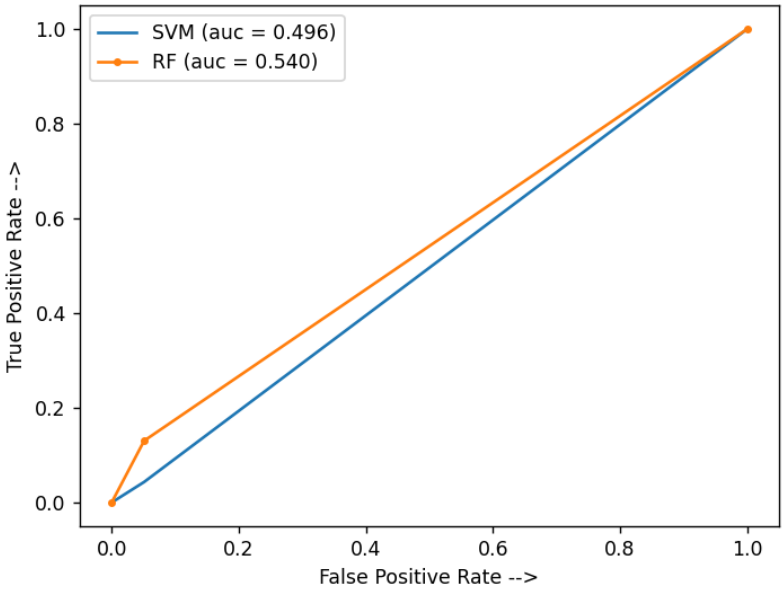
The following one is the confusion matrix for svm



The following one is the confusion matrix for rf



The following is the roc curve



Based on the f1-score, rf is slightly better than svm, while considering the accuracy, svm is slightly better than rf. Instead, with respect to the roc curve, rf is slightly better than svm since its auc is greater. But both the confusion matrices aren't good. In fact they quite well predict samples of class 0 but not the samples of the other classes. This is because the dataset is imbalanced (there are a lot of samples of class 0 and few samples of other classes). It is visible also looking at the precision, recall and f1-score table of the models (the highest values are only in the class 0 line).

So, I have to try to balance the dataset.

The technique I used was the random oversampling. It allows to duplicate samples in minority classes. In this way, there are 538 samples for each class (because the largest class is the 0 one with 538 samples).

In this case the results of both the models increases.

The accuracies are 0.8 for svm and 0.87 for rf.

The following are the precision, recall and f1-score

SVM table

	precision	recall	f1-score	support
0	0.610	0.538	0.571	186
1	0.592	0.541	0.565	185
2	0.782	0.931	0.850	173
3	0.972	1.000	0.986	172
4	1.000	1.000	1.000	172
accuracy			0.794	888
macro avg	0.791	0.802	0.794	888
weighted avg	0.785	0.794	0.788	888

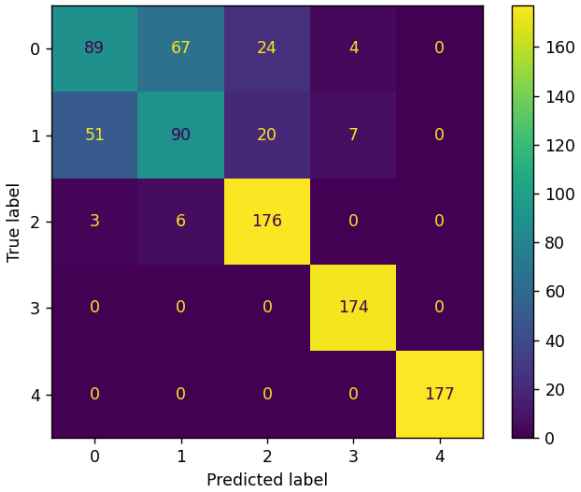
RF table

	precision	recall	f1-score	support
0	0.656	0.758	0.703	186
1	0.712	0.600	0.651	185
2	1.000	1.000	1.000	173
3	1.000	1.000	1.000	172
4	1.000	1.000	1.000	172
accuracy			0.866	888
macro avg	0.873	0.872	0.871	888
weighted avg	0.868	0.866	0.865	888

We can notice that precision, recall and f1-score are 1 for class 4 in the case of svm and for classes 2, 3 and 4 in the case of rf. It means that they are perfectly predicted.

Also, the confusion matrices are better than the previous ones.

For instance, here we have the confusion matrix of svm

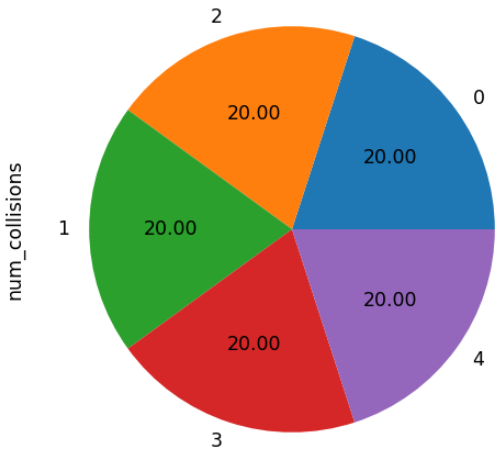


We can notice that samples of classes 3 and 4 are correctly classified.

The problem is that in the dataset there are only duplicate samples of this classes. Therefore, the models are overfitting. Moreover, I have balanced all the dataset (not only the training set). Also for this reason the models are overfitting.

Hence, I tried with balancing only the training set.

The percentages of the samples of each class in the training set is this (after random oversampling):



So the classes are equally distributed.

The results are the following ones:

```
SVM accuracy: 0.5242424242424243
RF accuracy 0.5363636363636364

SVM table
              precision    recall  f1-score   support

           0      0.554     0.856     0.673       174
           1      0.407     0.200     0.268       120
           2      0.000     0.000     0.000        28
           3      0.000     0.000     0.000         8

    accuracy                          0.524       330
   macro avg      0.240     0.264     0.235       330
weighted avg      0.440     0.524     0.452       330


RF table
              precision    recall  f1-score   support

           0      0.554     0.862     0.674       174
           1      0.466     0.225     0.303       120
           2      0.000     0.000     0.000        28
           3      0.000     0.000     0.000         8

    accuracy                          0.536       330
   macro avg      0.255     0.272     0.244       330
weighted avg      0.461     0.536     0.466       330
```

Differently from the case where I didn't balance the dataset, here at least some samples of class 2 are well predicted. Also in this case the rf f1-score is slightly higher than the svm one (same for accuracy), so rf is better.

So, I tried another technique to balance the dataset. I gave the classes weights. The class 0 (the one with most samples) has weight 1, class 1 has weight equals to the division between number of samples of class 0 and number of samples of class 1 and so on. In this way each class has a weight that is proportional to its number of samples in the dataset.

There are not so many differences on the results with respect to the case in which I didn't balance the dataset.

In order to increase the performances of the models, we can try different parameters with the gridsearch.

We know that the gridsearch uses cross-validation to measure performance of each set of hyper-parameters.

I implemented the gridsearch both for svm and rf. The gridsearch is used to find the parameters of the model that improve the results.

For svm, the most important parameters are:

- Kernel: what kind of kernel is used
- C: penalty parameter which represents misclassification or error term.
- gamma: defines how far influences the calculation of line of separation

In the case of imbalanced dataset, for svm I found that C=1, gamma=1 and kernel=linear are the best parameters.

For rf, the most important parameters are:

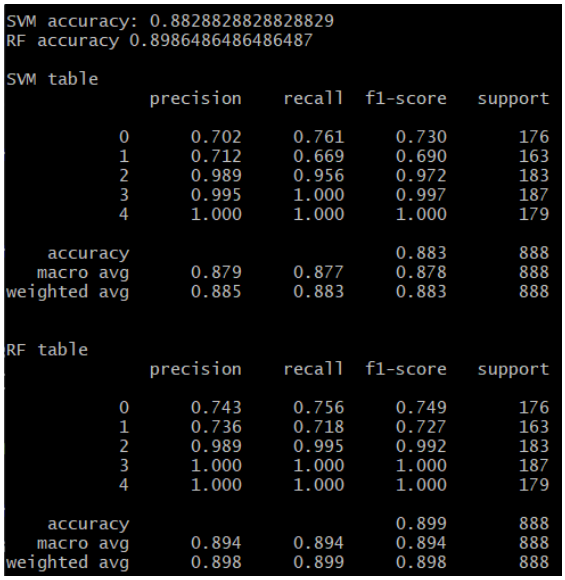
- n_estimators: number of trees in the forest
- max_features: max number of features considered for splitting a node

- max_depth: max number of levels
- min_samples_split: minimum number of data points placed in a node before the node is split
- min_samples_leaf: minimum number of data points allowed in a leaf node
- bootstrap: method for sampling data points

In the case of imbalanced dataset, for rf I found that bootstrap=true, max:depth=80, max_features=2, min_samples_left=3, min_samples_split=8, n_estimators=100 are the best parameters.

With these parameters, the performances of both the models slightly improve.

Also in the case of balanced dataset with random oversampling the performances of both the models improve with the parameters found by the gridsearch.



SVM accuracy: 0.8828828828828829
RF accuracy 0.8986486486486487

SVM table

	precision	recall	f1-score	support
0	0.702	0.761	0.730	176
1	0.712	0.669	0.690	163
2	0.989	0.956	0.972	183
3	0.995	1.000	0.997	187
4	1.000	1.000	1.000	179
accuracy			0.883	888
macro avg	0.879	0.877	0.878	888
weighted avg	0.885	0.883	0.883	888

RF table

	precision	recall	f1-score	support
0	0.743	0.756	0.749	176
1	0.736	0.718	0.727	163
2	0.989	0.995	0.992	183
3	1.000	1.000	1.000	187
4	1.000	1.000	1.000	179
accuracy			0.899	888
macro avg	0.894	0.894	0.894	888
weighted avg	0.898	0.899	0.898	888

So, I can say that we have five different cases:

- Normalized and imbalanced dataset: Based on the f1-score, rf is slightly better than svm, while considering the accuracy, svm is slightly better than rf. But both the confusion matrices aren't good.
- Normalized and balanced dataset (with random oversampling): the performances of both the models increases. Based on accuracy and f1-score, rf is better than svm. The problem is that the models are overfitting.
- Normalized and balanced training set (with random oversampling) with gridsearch: some samples of class 2 are well predicted. Based on f1-score and accuracy, rf is better.
- Normalized and imbalanced dataset with gridsearch: Based on the f1-score, rf is slightly better than svm, while considering the accuracy, svm is slightly better than rf.
- Normalized and balanced dataset (with random oversampling) with gridsearch: Based on accuracy and f1-score, rf is better than svm. The problem is still that the models are overfitting.

Finally, I think that it isn't so good to balance all the dataset (not only the training set) with random oversampling because of overfitting. So, although the results are not so better, we can

balance only the training set. Then, by using the gridsearch, we can say that based on f1-score (which is usually more useful than accuracy) rf performs better than svm.

Furthermore, since our dataset is quite large, rf is going to perform better than svm in general in terms if computational time. In fact, rf works faster than svm. If we add trees to the random forest, its computational time increases.

REGRESSION PROBLEM

With the regression problem, we have to consider the min_cpa. So, we have to take into account the features in order to predict a real value.

As in the classification problem, I read the dataset through pandas and prepared the dataset to be split. By dropping the columns 'min_CPA' and 'num_collisions', I obtained the features for the classification. By holding only the column 'min_CPA', I obtained a frame with the things we want to predict.

Initially I normalized only the features and the results were that the mean square errors were really high and both the r2-score were negative.

So, I normalize both the features and the thing we want to predict (min_CPA) obtaining values between 0 and 1.

Then I split the dataset in training set and test set (1/3 of data for the test set and 2/3 of data for the training set).

I used random forest regressor and polynomial svr (with polynomial kernel) as models.

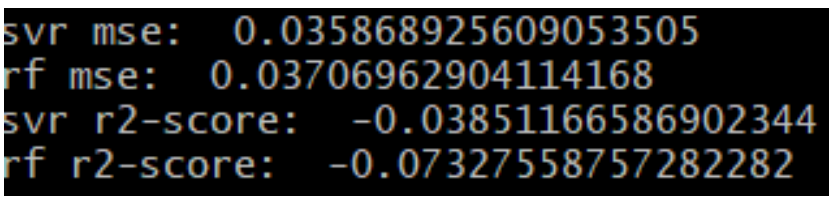
The metrics used to evaluate the models are the mean square error and the f2-score.

We know that:

- Mean square error is calculated by taking the mean of errors squared from data as it relates to a function
- F2-score is the weighted mean of precision and recall (it gives more weight to recall than to precision).

So, a model is performing well when its mean square error is low and its f2-score is high.

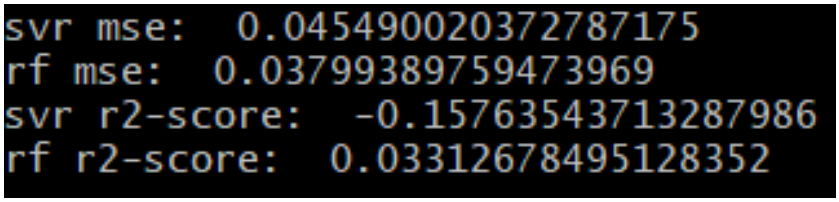
By training and testing the models, the results are the following ones:



This means that the models are not working well. Since the mean square error is lower in the case of svr and the r2-score is higher always in the case of svr, we can say that it performs slightly better than rf.

In order to slightly improve the results, we can do the gridsearch that returns the best parameters for the model.

With these parameters obtained through the gridsearch, the results are the following ones:



We can notice that at least the r2-score of rf is positive now. In this case, random forest is better.