SAPIENZA
UNIVERSITÀ DI ROMA

Machine Learning a.y. 22-23

Homework 1: Report

Hazem Dewidar - dewidar.1883881@studenti.uniroma1.it

December 8, 2022

1 Introduction

This homework requires us to implement two solution, both for classification and regression. In this report I'll go through the preprocessing of data, all the machine learning algorithms used, hyper-parameters tuning and finally, I'll compare the result.
I was given a dataset in which there were 35 features and the last columns were labels. The second last was for classification (there are 4 classes and they are unbalanced) and the last one is for regression. For this homework, I have leveraged the skilearn library and the unbalanced-learn library

1.1 Classifier

I've tried many classifiers such as:

- Decision tree classifier
- SVC
- Random Forest
- KNN

As far as concerned the label distribution, SMOTE was used to balance the dataset. In figure 1 we can see how there are very few samples of class 4 with respect to class 0.

The performance varies whether I put the SMOTE before the train split test or after the split (see section 3.1.1 and 3.1.2)

1.2 Regression

I regression, we've used the following machine learning techniques:

- Random Tree Regressor
- Linear regression

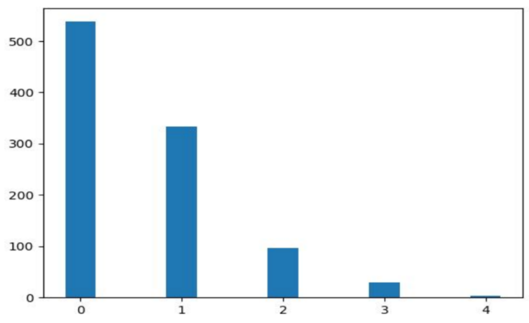

Figure 1: Data distribution

2 Models and data preprocessing

In this section I will discuss the different models that I've used and give a brief explanation to how they work.

Regression and classification are two types of supervised learning tasks in machine learning. Regression predicts a continuous values (for example house pricing given the size and other features) while classification predicts a discrete value. In this homework, the continues values for regression are the minimum closest point of approach (CPA) and the discrete values are total number of conflicts. In both regression and classification, the goal is to train a model on a labeled dataset, where the correct output values are known. The model can then be used to make predictions on new, unseen data.

2.1 Classification

For all the classifiers below, the dataset was preprocessed in the same way:

- Load it without the last two columns
- normalize it using the MinMaxScalar class of skilearn
- Use SMOTE. As said before, here we might use SMOTE before the train test split (so it's applied on all the dataset) or after (so it will be applied only to the training set). The performance will change drastically. SMOTE works by generating synthetic samples of the minority class that are then added to the original dataset. This has the effect of increasing the number of samples in the minority class, and balancing the dataset. The documentation of the library states that it should be applied only in the training set but it's interesting to see how performance varies.

2.1.1 Decision Tree

In scikit-learn, the DecisionTreeClassifier class is used to implement a decision tree for classification tasks. The DecisionTreeClassifier has several parameters that can be adjusted to improve the performance of the model, such as the maximum depth of the tree, the minimum number of samples required to split a node, and the criterion for splitting nodes.

2.1.2 SVC

In scikit-learn, the SVM algorithm is implemented in the SVC (Support Vector Classification) for classification task. SVMs are based on the idea of finding the hyperplane in a high-dimensional space that separates the most the two classes, or that minimizes the error for regression tasks. The distance from the hyperplane to the closest data points is called the margin, and the goal of the SVM algorithm is to maximize this margin. The hyperparameters of an SVC are:

- Kernel: This is the kernel function that is used to compute the similarity between the samples in the dataset. Common kernel functions include linear, polynomial, and radial basis functions.
- C: This is the regularization parameter that controls the trade-off between maximizing the margin and minimizing the error. A larger value of the regularization parameter will lead to a smaller margin, but a lower error.
- Gamma: This is a parameter that is only used with nonlinear kernels (e.g., the radial basis function kernel). It controls the curvature of the decision boundary, and can have a large impact on the performance of the model.
- Degree: This is a parameter that is only used with polynomial kernels. It controls the degree of the polynomial, and can have a large impact on the performance of the model. A high degree can lead to overfitting.

2.1.3 Random Forest Classifier

The random forest classifier is a popular ensemble learning algorithm that is used for both classification and regression tasks. It is a type of supervised learning algorithm that is based on the idea of training multiple decision trees on a subsample of the dataset and averaging their predictions to make more accurate and stable predictions. The model takes as input (during initialization) the following:

- The number of trees in the forest, which determines the ensemble size and the amount of averaging that will be done.
- Criterion: I've used entropy

One additional parameter would be max depth

2.1.4 KNN

It is a type of instance-based learning algorithm, where the model makes predictions based on the similarity of the input samples to the training data. In K-nearest neighbors (KNN), the training phase is used to store the training data. This data is used during the prediction phase, when the model makes predictions for a given input sample by finding the k-nearest neighbors in the training data and combining their target values by taking a majority vote. When making a prediction, we can specify how many neighbors we want to take the majority vote from and the distance metric

2.2 Regression

For the regression model, create a new dataset d (without the second last column) and then normalize it. Afterwards, I split d into the features x and the target value y. Finally, I split x and y into the training set and test set.

2.2.1 Linear Regression

In linear regression, the goal is to find the line (or hyperplane, in the case of multiple independent variables) that best fits the data. This is done by minimizing the sum of the squared errors between the predicted values and the true values. The resulting line (or hyperplane) can then be used to make predictions for new data. In scikit-learn, the linear regression algorithm is implemented in the LinearRegression class. This class provides a simple and efficient way to train and evaluate linear regression models on a variety of datasets.

2.2.2 Random Forest Regressor

See 2.1.3

3 Experiments

In this section I will show the plots and the results obtained. On some models (SVMs for example) I've applied a grid search for tuning the hyper-parameters.

3.1 Classifier

Let's distinguish two runs of experiment: one with SMOTE applied on the entire dataset and one with SMOTE applied to just the training set.

3.1.1 SMOTE applied only to the training set

The best **KNN** configuration is:

- weights: uniform
- neighbor: 1
- algorithm: auto
- leaf_size = 10
- p: 1

This configuration has produced an accuracy of 0.40 (+/- 0.04) using k fold validation.

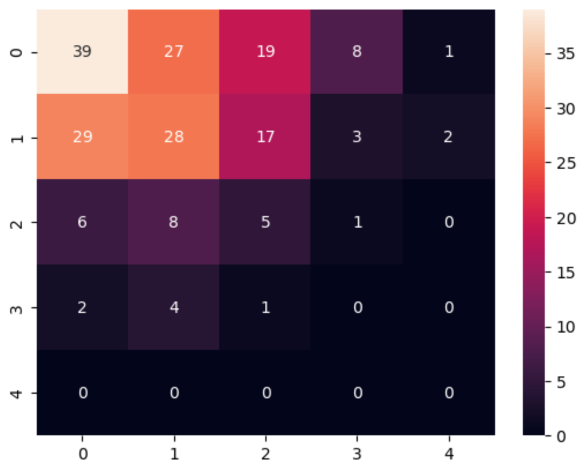

Figure 2: KNN Confusion Matrix

Below there is the output of the classification report

```
              precision    recall  f1-score   support

           0       0.51      0.41      0.46        94
           1       0.42      0.35      0.38        79
           2       0.12      0.25      0.16        20
           3       0.00      0.00      0.00         7
           4       0.00      0.00      0.00         0

    accuracy                           0.36       200
   macro avg       0.21      0.20      0.20       200
weighted avg       0.42      0.36      0.38       200
```

The best **Random Forest** configuration is:

- n_estimators = 30

This configuration has produced an accuracy of 0.51 (+/- 0.03) using k fold validation.

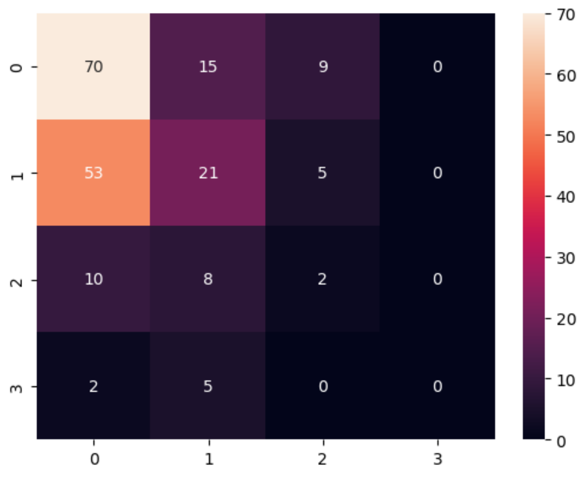

Figure 3: Random Forest Confusion Matrix

Below there is the output of the classification report

```
              precision    recall  f1-score   support

           0       0.52      0.74      0.61        94
           1       0.43      0.27      0.33        79
           2       0.12      0.10      0.11        20
           3       0.00      0.00      0.00         7

    accuracy                           0.47       200
   macro avg       0.27      0.28      0.26       200
weighted avg       0.43      0.47      0.43       200
```

The best **SVC** configuration is:

- kernel: rbf
- gamma: 0.5
- C: 0.5

This configuration has produced an accuracy of 0.54 (+/- 0.01) using k fold validation.

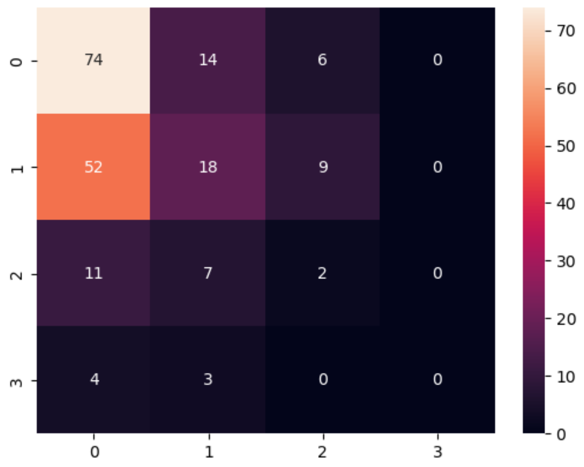

Figure 4: SVC confusion matrix

Below there is the output of the classification report

```
              precision    recall  f1-score   support

           0       0.52      0.79      0.63        94
           1       0.43      0.23      0.30        79
           2       0.12      0.10      0.11        20
           3       0.00      0.00      0.00         7

    accuracy                           0.47       200
   macro avg       0.27      0.28      0.26       200
weighted avg       0.43      0.47      0.42       200
```

As far the **decision tree classifier**, I've obtained an accuracy of 0.41 (+/- 0.05) using k fold validation.

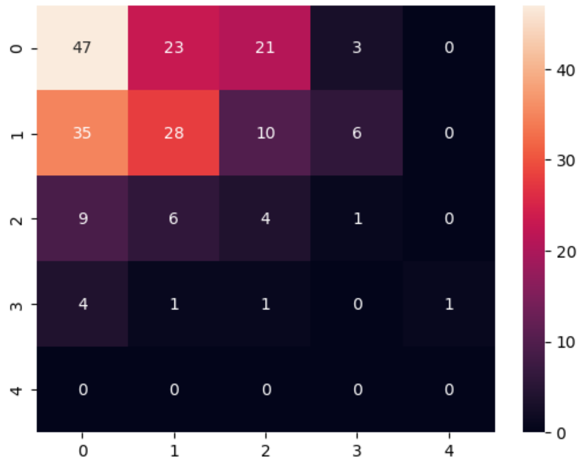

Figure 5: decision tree confusion matrix

Below there is the output of the classification report

```
              precision    recall  f1-score   support

           0       0.49      0.50      0.50        94
           1       0.48      0.35      0.41        79
           2       0.11      0.20      0.14        20
           3       0.00      0.00      0.00         7
           4       0.00      0.00      0.00         0

    accuracy                           0.40       200
   macro avg       0.22      0.21      0.21       200
weighted avg       0.43      0.40      0.41       200
```

3.1.2 SMOTE applied the entire dataset

The best **KNN** configuration is:

- weights: uniform
- neighbor: 1
- algorithm: auto
- leaf_size = 10
- p: 1

This configuration has produced an accuracy of 0.82 (+/- 0.02) using k fold validation.

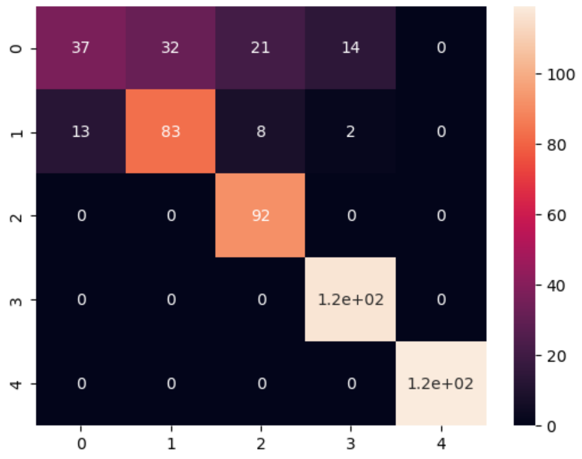

Figure 6: KNN Confusion Matrix

Below there is the output of the classification report

```
              precision    recall  f1-score   support

           0       0.74      0.36      0.48       104
           1       0.72      0.78      0.75       106
           2       0.76      1.00      0.86        92
           3       0.88      1.00      0.94       117
           4       1.00      1.00      1.00       119

    accuracy                           0.83       538
   macro avg       0.82      0.83      0.81       538
weighted avg       0.83      0.83      0.81       538
```

The best **Random Forest** configuration is:

- n_estimators = 90

This configuration has produced an accuracy of Accuracy: 0.86 (+/- 0.02) using k fold validation.

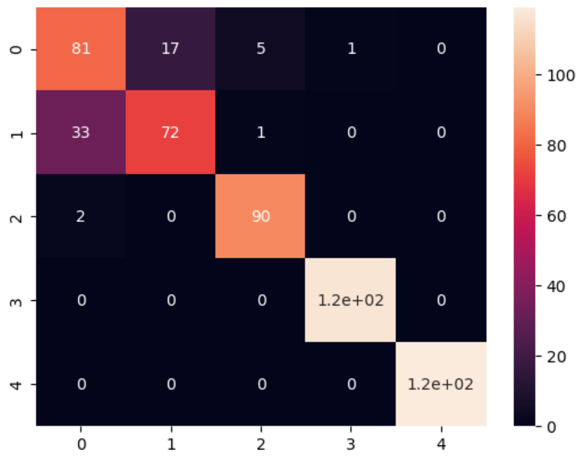

Figure 7: Random Forest Confusion Matrix

Below there is the output of the classification report

```
              precision    recall  f1-score   support

           0       0.70      0.78      0.74       104
           1       0.81      0.68      0.74       106
           2       0.94      0.98      0.96        92
           3       0.99      1.00      1.00       117
           4       1.00      1.00      1.00       119

    accuracy                           0.89       538
   macro avg       0.89      0.89      0.89       538
weighted avg       0.89      0.89      0.89       538
```

The best **SVC** configuration is:

- kernel: poly
- gamma: 0.5
- C: 0.3
- degree 5

This configuration has produced an accuracy of 0.87 (+/- 0.06) using k fold validation.

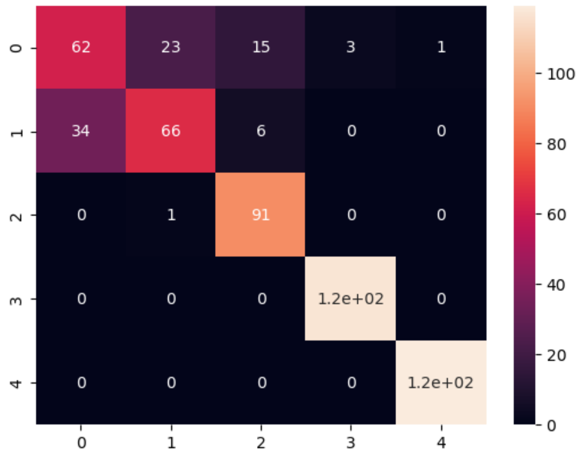

Figure 8: SVC confusion matrix

Below there is the output of the classification report

```
              precision    recall  f1-score   support

           0       0.65      0.60      0.62       104
           1       0.73      0.62      0.67       106
           2       0.81      0.99      0.89        92
           3       0.97      1.00      0.99       117
           4       0.99      1.00      1.00       119

    accuracy                           0.85       538
   macro avg       0.83      0.84      0.83       538
weighted avg       0.84      0.85      0.84       538
```

As far the **decision tree classifier**, I've obtained an accuracy of 0.74 (+/- 0.02)) using k fold validation.

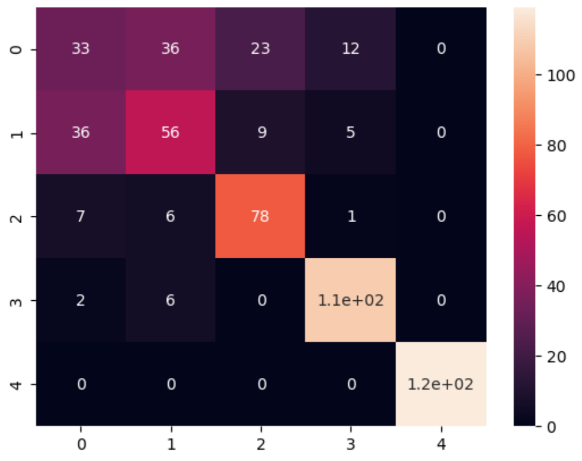

Figure 9: Decision tree confusion matrix

Below there is the output of the classification report

```
              precision    recall  f1-score   support

           0       0.42      0.32      0.36       104
           1       0.54      0.53      0.53       106
           2       0.71      0.85      0.77        92
           3       0.86      0.93      0.89       117
           4       1.00      1.00      1.00       119

    accuracy                           0.73       538
   macro avg       0.71      0.73      0.71       538
weighted avg       0.72      0.73      0.72       538
```

3.2 Regression

The best **SVR** configuration is:

- kernel: poly
- gamma: 0.3
- C: 0.3
- degree 2

It has an r2 score of 0.11.

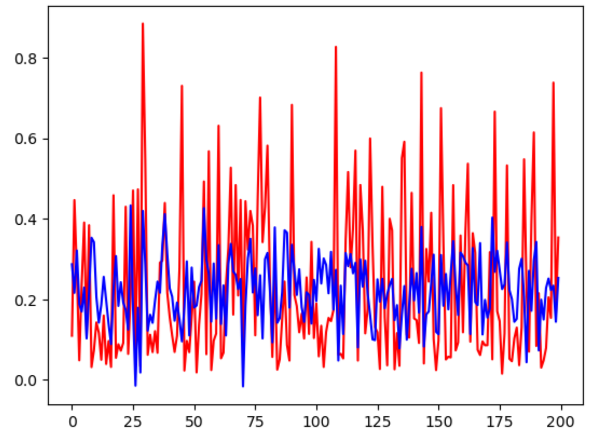

Figure 10: SVR. In red the real values. In blu the predicted values

The **linear regression** has an r2 score of -0.04. That means that performs worse than a model who always gives the same output

Finally, the **random forest regressor** with 250 esimators has a r2 score of +0.04

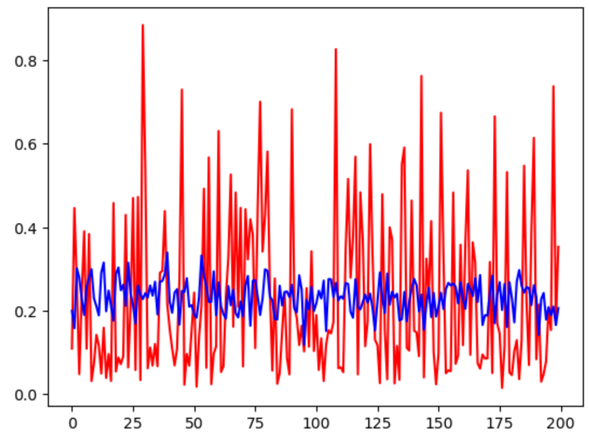

Figure 11: Linear Regression. In red the real values. In blue the predicted values

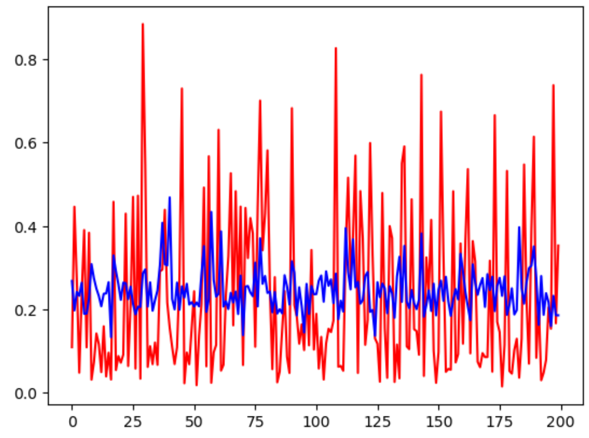

Figure 12: Random Forest Regressor. In red the real values. In blu the predicted values

4 Conclusions

To summarize, the best models in regression and classification, on this tasks, were Support Vector machines. The preprocessing of the dataset was really important since it was heavily imbalanced (the class 4, for example, had really few samples in the dataset). Sometimes, ore models, performs really bad (especially in the case of regression) where we obtain a low r2 score. In the case of classification, when we apply SMOTE only on the training set, we obtain around 54% accuracy. For validation, I've used the k fold technique.