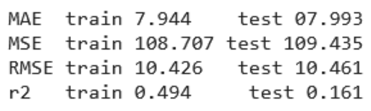
Machine Learning: regression, deep networks and dimensionality reduction

Weight Predictions

Students:
Guglielmo Colombo
Irene Rigato 18232100

First semester 2021/2022

Contents



```
MAE  train 7.944    test 07.993
MSE  train 108.707 test 109.435
RMSE train 10.426   test 10.461
r2   train 0.494     test 0.161
```

Figure 1: Linear Regression performed on raw data to show overfitting

1 Introduction

Machine learning (ML) is the study of computer algorithms that can improve automatically through experience and by the use of data. Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions. The aim of this project was to find the best machine learning model able to predict the weights, given as training data a set of independent features characterizing the behaviours or characteristic of the sample person.

2 Data Engineering

At a first sight, the provided dataset was very small, of only 250 samples for each feature. Building a model on such small dataset would likely lead to overfitting. As a matter of fact applying a *Linear Regression* on raw data the obtained scores make us think that model encounters overfitting. As shown in figure 1 it can be seen as an example that the R^2 indicator on the training set is much bigger with respect to the test set one. To limit this problem, that has even a huge influence on small dataset, the following approaches were used:

- outliers removal,
- data standardization,
- data augmentation,
- data cross validation.

In first approach data were observed and neither NA nor duplicates were found. Indeed, the presence of NA does not allow to compute the prediction; duplicates instead does not provide any information making the computation heavier.

Furthermore, looking at the feature types, in order to be able to work with all the data it is necessary to convert the object data type into numeric ones, otherwise data cannot be standardized and models will not work if not with numerical values. The *factorize* function was used for this scope, which operates as a label encoding. Another possible approach to deal with categorical data is the One Hot encoding, which consist in splitting the relative column of a categorical data in many columns as the number of different categories that could take the feature, then for each sample the columns relative to the categorical value of the sample will be set to one. However in our dataset the data are quite correlated between each other, and the use of One Hot enconding lead to multicollinearity, especially with a model like Liner Regressor. To prepare data it is also useful to remove the outliers; indeed their presence can lead to inaccurate predictions. Outliers are extreme values that fall a long way outside of the other observations.

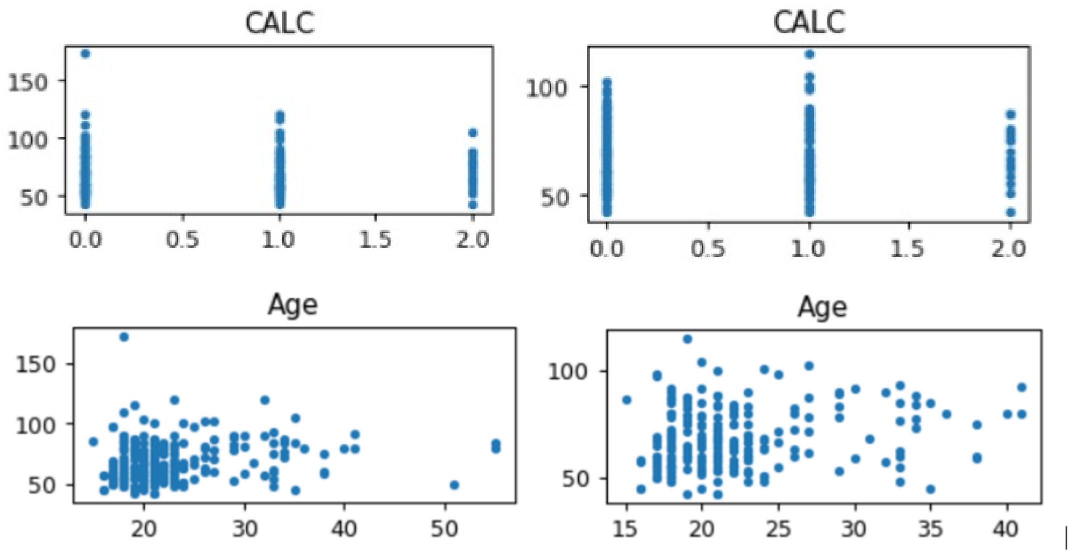


Figure 2: Scatterplot before and after outliers removal with zscore on categorical and regression feature

In a small dataset, the impact of an outlier can be much greater, since it will have a heavy weight for the model.Two ways of treating them were approached: the *quantile* and the *zscore* methods. To compare the two, the *scatterplot* was used and as a result it was observed that the quantile method does not perform well in presence of many categorical variables so the zscore one was chosen. The performance of *zscore* can be seen in figure 4 on the *CALC* feature.

Subsequently, looking at the plot it was decided to discard the *SMOKE* variable since after the outliers removal all data fell into one category, so the informative content is null.

Then it is better to standardize data in order to work with uniform unit of measure between features; indeed comparing data with the same criteria and different scales can lead to misleading conclusions. So, we have chosen to use *minmax* scaler method because we have already removed outliers and this method in comparison with z-index one leads to better result even if the distribution is not a normal one.

2.1 Feature Selection

While removing outliers consists of deleting rows from the dataset, feature selection consists of deleting columns that do not contribute to the prediction. The best features according to the target where selected basing on the amount of correlation computed with the Pearson method. Then, for the target *weight* the optimal number of features was found to be seven after have been testing the efficiency on a regression model.

2.2 Data augmentation

Since the size of the dataset is limited (250 samples), in order to obtain better results and a model that avoids overfitting, we tried to augment the dataset. SMOTE (Synthetic Minority Oversampling TEchnique) consists of synthesizing elements for the minority class, based on those

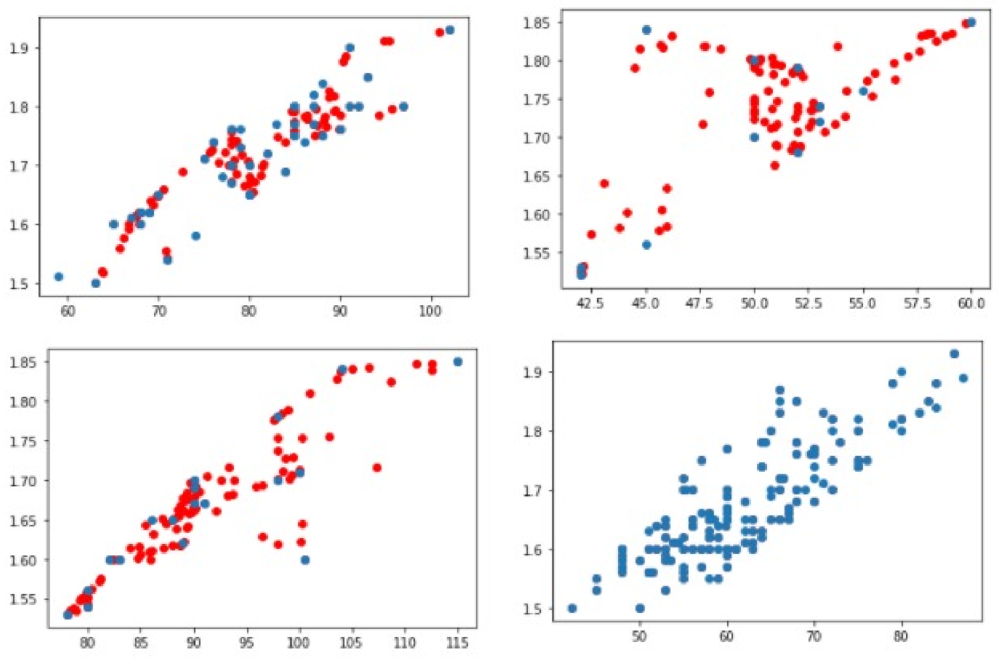


Figure 3: Data Augmentation scatter plot

that already exist. It works randomly picking a point from the minority class and computing the k-nearest neighbors for this point. The synthetic points are added between the chosen point and its neighbors. More specifically was used the SMOTENC library, which is more suitable for our case study since most of the features are categorical and SMOTENC takes into account the categorical features, creating new samples with a classification method and not a regression. Before augmenting the number of samples, the dataset was splitted between the train and test part. The augmentation has regarded only the training part of the dataset; this was used to fit the model that afterwards was evaluated on the original dataset. However, for our dataset it was still not possible to apply the method because it requires to have a classed target, which are not the "weights" we have as target. That's why before applying SMOTE, the BMI was computed, and then divided into the four classes for each range. In this way we had more than one feature for each class and the SMOTE could have been applied, even balancing a strong unbalanced dataset, since very few features have as target an extreme value of BMI (underweight or obese). However, the obtained results were quite poor and worse than the ones obtained with the original dataset (an RMSE error 5 points worse) and a score regression for a logistic regression model of 0.26, very high compared to the one obtained with other models. The PCA(2) visualization of the dataset augmented with SMOTE in Figure 4 clearly show the wrong distributions of the new values. This can be seen even from the plots of the new values produced for the weights produced for each BMI class. For a BMI equal to zero, which is the class with less samples in the original dataset (top right figure), the new produced points are quite far from the original points (red points vs. blu points). Another data augmentation method we decided to apply is the tabGan library [1]. Usually GANs are used in the realistic image generation. However, they can be applied in tabular data generation, and this is the purpose they are used for in this library. In this case the new datapoints produced are more similar to the original ones, so we consider this augmented data set for our model analysis. As can be seen from the plots of the

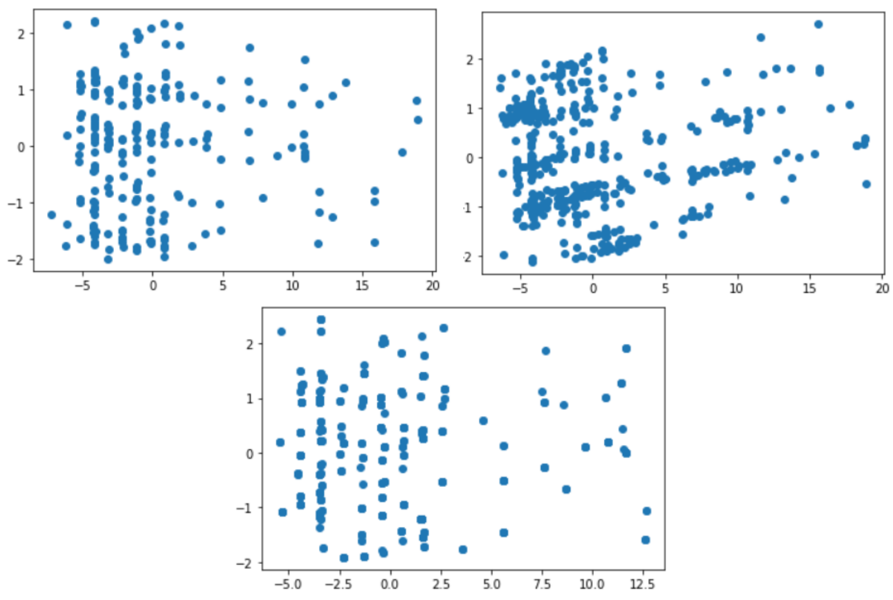


Figure 4: Original Data PCA(2), Data Augmented with SMOTE PCA(2), Data Augmented with TabGan PCA(2)

data features (excluded the weight) after applying a PCA with two components (to allow an easy data visualization) the distribution of the original data and of the augmented dataset are very similar.

3 Model

All the following analysis where conducted with a train test split of 0.3, since with this proportion we have obtained good score overall and the function scoreregression has enough sample to avoid the *no true samples* error. Indeed using a too low percentage of points for the test set (less than 0.15) since BMI class target are not balanced, happened that for the extreme case of BMI no samples are selected. For all the models we implemented a GridSearch to iterate over all the possible combination of parameters of a single model, using a cross validation with 5 splits, and the Negative-Root-Mean-Square-Error as scoring function. We opted for RMSE as scoring function since it is the loss function more similar to the non aligned loss of "scoreregression" used to chose the best model. The models we used to perform the analysis are:

- Linear Regression = this is the most simplex model, because of that we are expecting good performances in terms of avoiding overfitting.
- KNN = the parameter of the grid we used are the number of K neighbours. The best values obtained for the parameters are:

  'algorithm': 'auto' 'N neighbors': 20 'weights': 'uniform'

- Multi Linear Perceptron = we used the MLP Regressor method implemented in the sklearn library, with two hidden layers, tested with 20 and 40 sizes for hidden layer neurons. The best values obtained for the parameters are:

'activation': 'relu' 'alpha': 0.001 'hidden layer sizes': (40, 40)
'learning rate': 'constant' 'max iter': 400 'solver': 'lbfgs'

- SVM = trained with a non liner kernel, as in our dataset the distribution of data is not linear The best values obtained for the parameters are:

  'gamma': 'auto' 'kernel': 'poly'

- Random Forest Regressor = Random Forest limits the greatest disadvantage of Decision Trees, since it overfits less due to subset and feature randomization; having a small dataset is the reason we decided to implement it. And is a valid model for value prediction.

3.1 Analysis on weight target, no data augmentation

In figure 5 are shown the performance with the various error metrics: in this case to choose the best model we have given a look overall the scoring and at the scoreregression concluding that the Linear Regression is the model that performs better, with a scoreregression of 0.147. It is interesting to notice looking at the R^2 metric all the value suffer of a general decreasing due to better performance on the training set with respect to the test set; since this phenomena is expected for our dataset, the model that decreases less has an higher likelihood to be the chosen one. On the contrary, we have to keep far from the model that has great difference between the train and the test metrics because it may has incurred in overfitting. Above the result of score regression for each of the tested model:

Linear Regression: 0.147
Multi Linear Perceptron: 0.191
K Nearest Neighbours: 0.246
Support Vector Machine: 0.236
Random Forest: 0.209

3.2 Analysis on weight target, data augmentation

In the plot 6, we can see the performances of the model after a data augmentation with *Tabgan Sampler*. Looking at the graph it can be seen that the more robust model is linear regression again: the R^2 index is quite invariant in the train and test sets and this can be noticed also for the other metrics. This is not true for the other models, as for example *KNN* that presents very small errors on the training set and huge on the test one, due to the model overfitting on the training test. Indeed, the scoreregression for the linear regressor is the second lowest one, but we do not look much at the other models since them all has a bad generalization with respect to the linear regressor one. Above the result of score regression for each of the tested model:

Linear Regression: 0.143
Multi Linear Perceptron: 0.151
K Nearest Neighbours: 0.229
Support Vector Machine: 0.193
Random Forest: 0.138

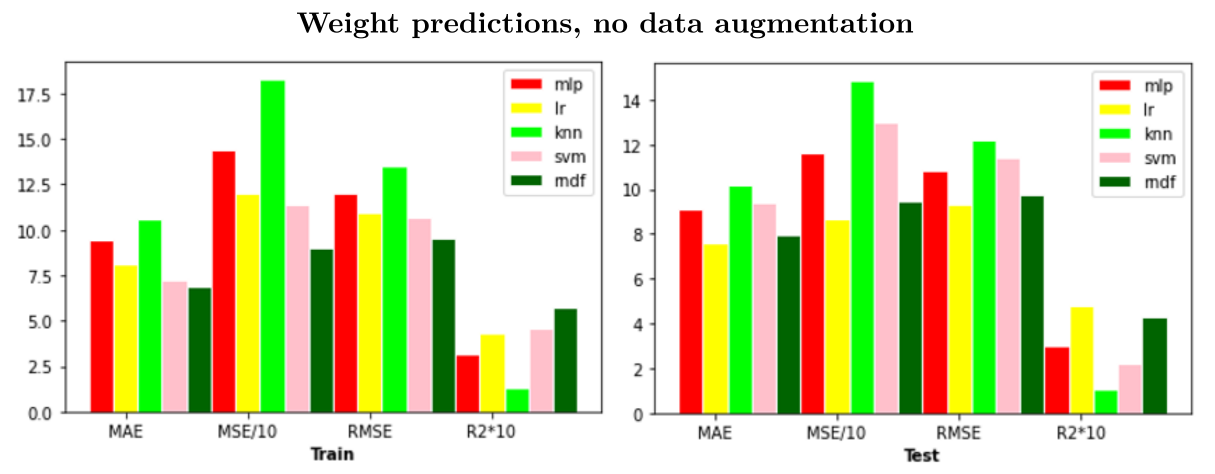


Figure 5: On the left errors indexes for each model basing on the training set; on the right errors indexes for each model basing on the test set.

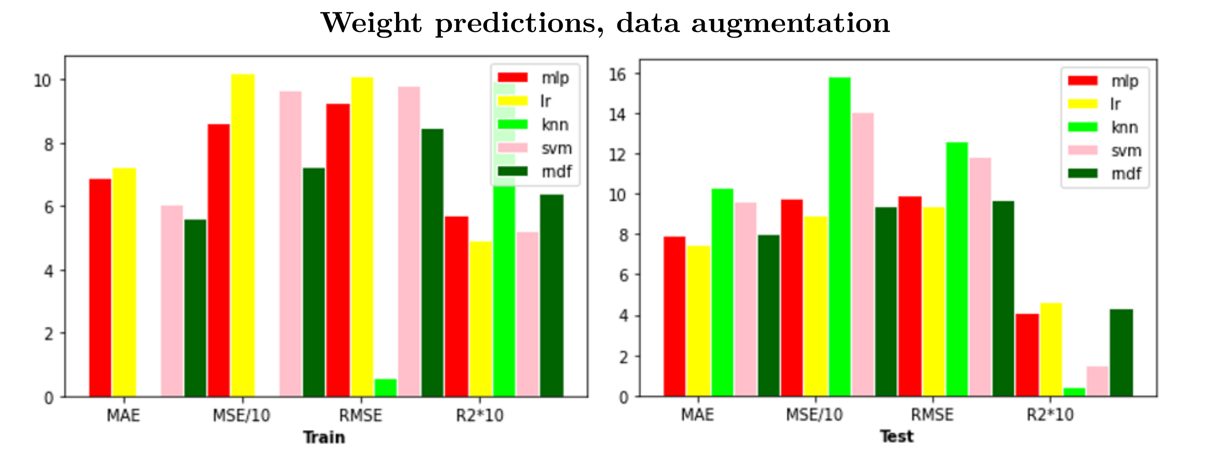


Figure 6: On the left errors indexes for each model basing on the training set; on the right errors indexes for each model basing on the test set.

Looking at these performance and comparing with the one of before the model used to produce prediction was the Linear Regressor trained in the augmented dataset.

3.3 Analysis on BMI target, no data augmentation

As suggested, we have tried to predict the BMI in place of the weight. At first sight can be seen that all the metrics and the scoreregression have lower values with respect of before and this is probably due to the fact that the predictions are made in a narrower range ($[13, 40]$) versus the wider range of the weights $[42, 173]$.

According to this analysis the most efficient model appear to be the Multi Layer Perceptron Regressor, both for the MAE, MSE, RMSE metrics and R^2, that in this case is even higher on the test set with respect to the training set. This performance seems quite suspicious so for the moment we take this model with a grain of salt. The minimum scoreregression is owned by the MLPRegressor and Linear Regressor, with a score of 0.179. Conducting the same analysis, the

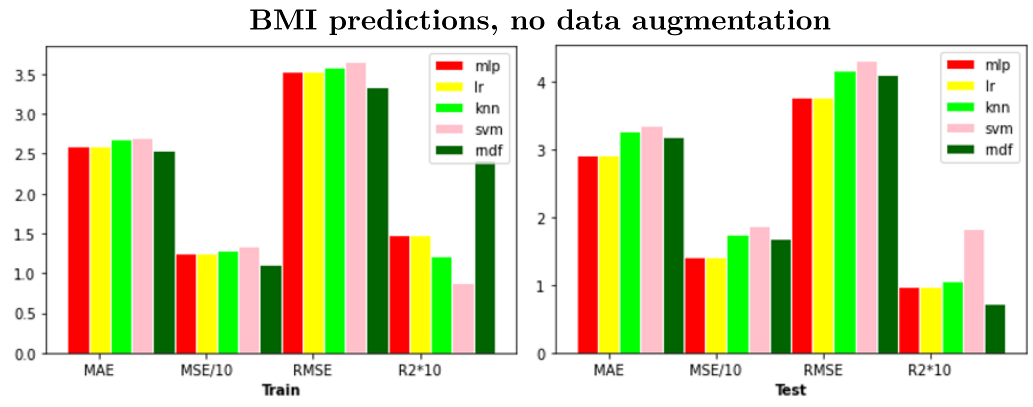
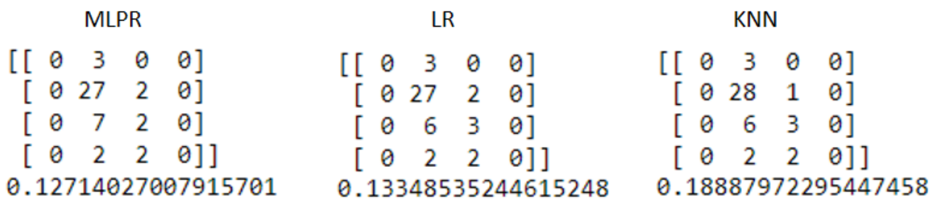


Figure 7: On the left errors indexes for each model basing on the training set; on the right errors indexes for each model basing on the test set.



```
MLPR                     LR                       KNN
[[ 0  3  0  0]           [[ 0  3  0  0]           [[ 0  3  0  0]
 [ 0 27  2  0]            [ 0 27  2  0]            [ 0 28  1  0]
 [ 0  7  2  0]            [ 0  6  3  0]            [ 0  6  3  0]
 [ 0  2  2  0]]           [ 0  2  2  0]]           [ 0  2  2  0]]
0.12714027007915701      0.13348535244615248      0.18887972295447458
```

Figure 8: Confusion matrices and relative scoreregression.

second model that has best performances is the Linear Regressor, but it is also interesting to notice that the model that remains R^2 coherent is the *KNN*.

The figure 8 displays the models which are able to better classify the classes: first we find the *KNN* with the 68% of correct classifications but with the highest scoreregression. This can be explained because we have interpreted the result as a classification task and not a regression one, so even if the error committed on the values is higher with respect to the others probably the range within these values were predicted was more accurate. Going deeper in the understanding of these confusion matrices, it can be noticed that the errors are committed for the largest part on the classes found on the edges, so the underweight and obese class. This phenomena can be explained by the fact that we have less samples for that classes and so the model is less trained to predict those. As a counter prove, the most common class is the best predicted one.

4 Conclusion

In this report we evaluated several popular machine learning algorithms to accurately predict weights. Issues related to a small dataset for training has been considered and dealt with data preprocessing and data augmentation. For robustness, a sample of unaltered data is used to test the prediction. The obtained results are far from being optimal, especially from the analysis of confusion matrixcs, but still encouraging. Most misclassification errors of the BMI classes are due to unbalanced BMI status of the samples of our dataset. How ever balancing the dataset with a the SMOTE metohd did not provide better results. Taking this into account and looking overall the metrics and the scoreregression the model that performs best according to our analisys was the Linear Regression model trained on the augmented dataset, according to accuracy of the prediction and to the robustness of the model. This can be explained since this model is a simple one, so has a good generalization property and few parameters to be chosen.

References

[1] GAN-for-tabular-data : https://github.com/Diyago/GAN-for-tabular-data