LELEC2870

Project: Predicting a person's weight based on eating habits and physical activities

Laurent Louis 6806 17000 louis.laurent@student.uclouvain.be
Robert Thomas 4845 1800 t.robert@student.uclouvain.be

December 19, 2021

1 Introduction

This is the report for the project from the *LELEC2870 - Machine Learning: regression, deep networks and dimensionality reduction* course. The goal of this project was to predict the BMI (Body Mass Index) of people based and their eating habits and physical activities. The implementation was done in Python using mainly the *scikit-learn* python library. We built different models that predict results while putting more emphasis on the class-balanced accuracy than the total accuracy.

2 Preprocessing

2.1 Data visualization

First, to get a feel for the data, and as advised in multiple of our courses or in the practical sessions, we printed a scatter plot to visualize the dataset. What we can notice is that the persons represented in the samples tend to be quite young, to be of average height and to have a normal weight. This is what prompted us to try putting weight sampling on the less represented features. (see "Data augmentation>Sample weights balancing")

We noticed (in relief) that there is no noticeable clusters of samples, which is positive since if there were, they could lower some of our *K-Fold* splits performances.

Another visualization that we performed was a heatmap that indicates correlation between features, and between features and target. From the results of the heatmap, we decided to perform feature selection (see "Feature selection").

2.2 data standardization

This was one of the first preprocessing step we performed, we used the *sklearn.preprocessing* module, and especially the *LabelEncoder()* method. It changed all categorical values into floats to standardize our dataframe and to transform values from different features into the same shape. At the same time we decided to make them consistent, so every time there was a no, it became a zero, while a yes would become a one. After some research, we deduced that we had to transform categorical values into numbers simply because the machine learning model we use requires number to function, but the idea is the same as if the data was categorical values. However, this method could introduce inaccuracies as the model could consider, for four labels that would have been transformed to 1, 2, 3, 4 that the 1 is closer to the 2 than to the 4. That is also why we tried to make for instance the "Walking" label closer to "Biking" than to "Automobile" in the MTRANS feature.

2.3 Feature selection

As explained above, the correlation heatmap made us try feature selection. To do so, we removed features that were under a certain threshold of correlation with the target *weight*. We of course checked the impact of removing each feature independently, of removing different subsets, and removing them all. The results we obtained can be seen in the table below. We also checked for features highly correlated between them, but we did not found any correlation worth removing one feature.

We concluded that the features: FAVC, FCVC, NCP, CAEC, SMOKE, SCC, FAF, TUE and CALC are not helping gaining information or even worse, they can have a negative impact on our predictions. As seen in the course, having less feature is beneficial in general because it allows the dataset to be smaller, to be trained faster, but mainly it makes it so that less data has to be collected for enabling the model to be used. That is why when a feature has a neutral impact on the prediction power of the model, it is best to remove it.

2.4 Outliers removal

We noticed there were at least one sample point that was an outlier from the visualization alone (the sample with a weight over 200kg), so we decided to perform outlier removal. To do so we simply computed the z-score of each sample point and removed all of those who where inferior to three. In total, we removed six samples from the dataset, which is now composed of 244 samples. We again checked the performance of the baseline models with and without the outliers removal step, as can be seen in the table below.

2.5 Results overview

	Linear	KNN	MLP	RF
Base models	0.1015	0.1913	0.1692	0.1711
Without Feature Selection	0.0978	0.1967	0.1828	0.1539
Without Outliers Removal	0.1139	0.2033	0.1807	0.1636
With Oversampling	0.1213	0.2077	0.1973	0.2370

Table 1: Score with the given function using or not techniques of preprocessing

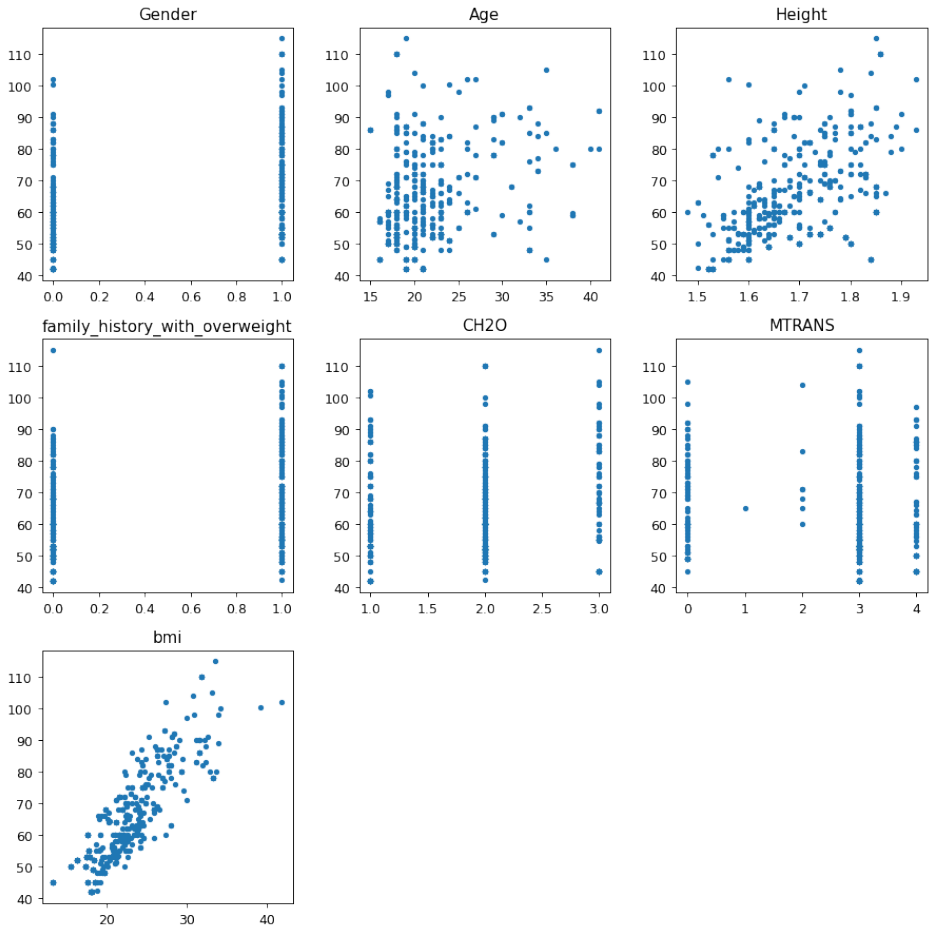

Figure 1: Scatter plot of the data after preprocessing

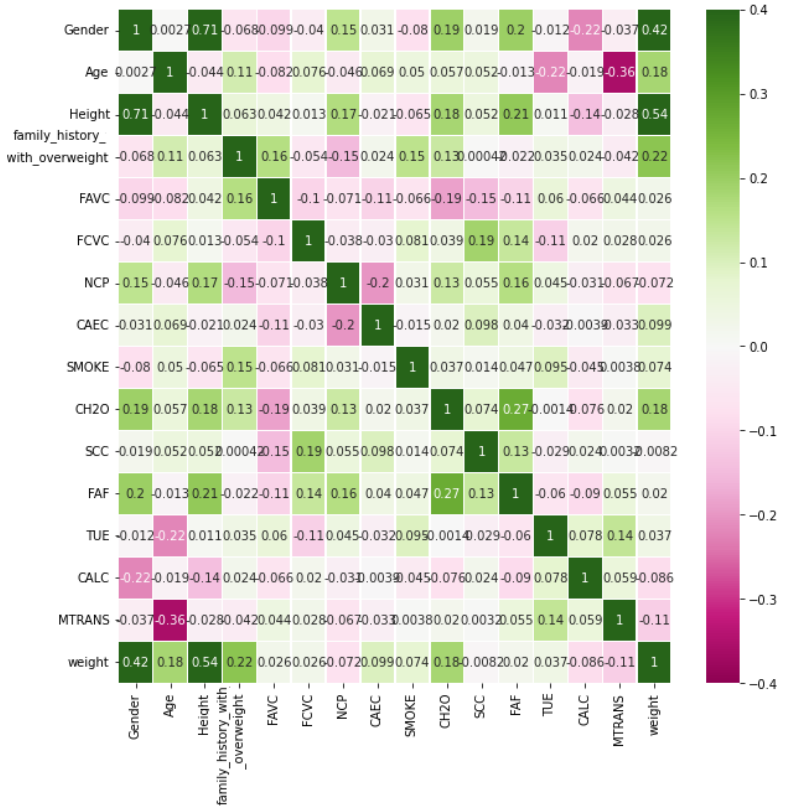

Figure 2: Correlation heatmap of the data before preprocessing (Excuse us for the cluttering of the values)

Figure 3: Correlation heatmap of the data after preprocessing

3 Data augmentation

3.1 Sample weight balancing

We tried doing weight balancing for rarely seen samples that we identified with the visualization. We put more weight on the samples of people that were a little bit older or younger (less than 15 or more than 37), that are very tall or very small (less than 155cm or more than 185cm), and people that used walking or public transportation as their usual mean of transport (MTRANS feature). The results were however disappointing since there was no score difference with the baseline model.

3.2 Class imbalance

A very important thing when building a classifier is to know exactly what your goal is. Here, the goal is not to have the best absolute score, but to have a classifier that has a good score for each BMI class. Since the overwhelming majority of samples have a normal BMI, a trivial classifier that would always return "normal bmi" would fare quite well. However in this project we have to do an effort to make sure even the minority classes (here underweight, overweight and very overweight) are still predicted by our classifier.

It is in this spirit that we tackled the problem of class imbalance: in each split of the *F-fold validation* (see below "Model and validation > Model evaluation") we tried "oversampling" the samples belonging to the minority classes. In other words, we simply duplicate the samples of the minority class in the training set so that all the classes are represented by approximately the same number of samples. It is to note that since the BMI is not present by default in the dataset, we computed and added a feature "BMI" that we use to perform the oversampling, and then remove before the model is fitted. This idea of class imbalance comes from the discussions on Moodle, the project statement and the paper that uses this dataset to perform a similar task to ours (However, we did not read the entirety of the paper in details, maybe it was a mistake and we missed other interesting ideas to improve our classifier).

Error The biggest mistake we did during this project is without a doubt linked to oversampling. What we first did was to oversample the whole dataset before dividing the dataset into a training and test part. It naturally caused the duplicated samples from the minority class to be present both in the training and the test set, which made reach unusually good regression scores. Our best score of 0.0305 was reached with a random forest regressor. We felt something was wrong because reaching such a good score did not seem realistic, but it still took us some days before finally finding the problem.

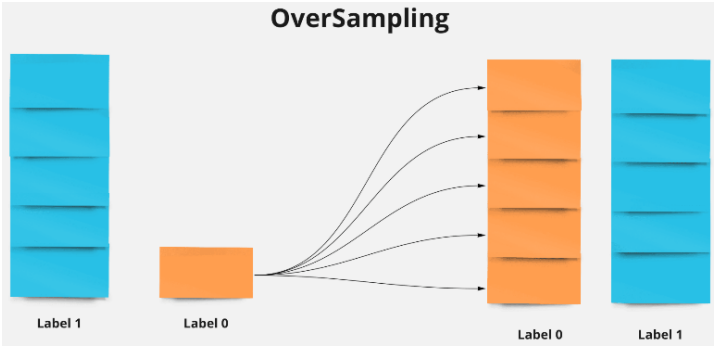

Figure 4: Image explaining the concept of oversampling

4 Model and validation

4.1 Model evaluation

The score used is the one provided in the project statement, and has to be minimized to obtain the best possible model.

To test all our models, we perform a 10-fold cross-validation where the model is fitted on 90% of the training set, before being evaluated on the remaining 10%. During the cross-validation, we consider the regression score and not the MSE because we want to reach a better regression score (which means a better score for each weight class), not a better absolute score, it therefore makes sense to directly test for the metric of interest.

For each split, the score is saved, so we can compute our final score as the mean of all the split's score at the end of the validation.

4.2 Candidates & model selection

As asked in the project statement, we have the following models: a linear regressor (LR), a multilayer perceptron regressor (MLP), and a K-Nearest Neighbors regressor (KNN). As extra model, we have a random forest regressor (RF).

After performing a grid search on most models (LR has almost no hyperparameters to tune), we simply selected the one with the highest class-balanced score. We therefore selected our baseline, the LR to perform the final prediction. Below is a bar graph representing the regression score of our different models before and after they were optimized with a grid search.

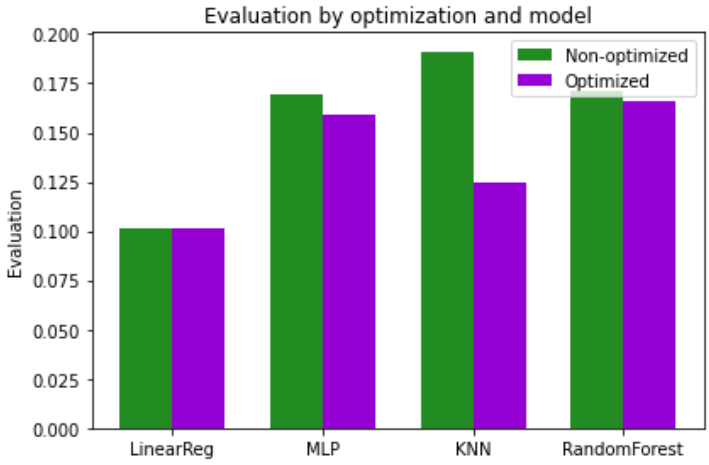

Figure 5: Bar graph representing the regression scores of our different models before and after performing a grid search

4.3 Best model

In this section, we discuss a possible explanation of why the LR model was the best at the BMI prediction task, and we provide a confusion matrix of this model.
The LP model is better when we use it to find a linear relationship between a few features and in our case, we have the feature *Height* that is highly correlated to the target *weight*. Since they linearly correlate well, it could be one explanation of why the LR model performs better than the others.

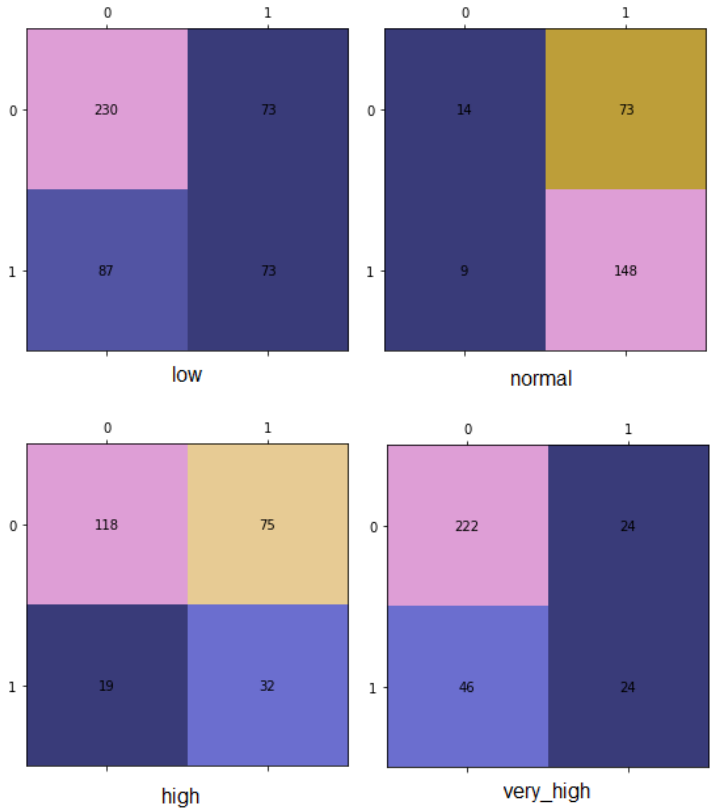

Figure 6: Confusion matrix of the LR model

5 Prediction

We are asked to predict the performance of our model on the second half of the dataset. Since the data comes from the same dataset, we are not confronted to a lowered accuracy due to different measurement/interviewing techniques (i.e. the way the data was collected), nor in a variation of the target population (here the data comes from South America, and if you had a dataset coming from the United States, there could be a significant drop in accuracy). This is because cross-validation is good but not perfect to minimize over-fitting, and will not perform well if the data we have is different from the data we are trying to predict.

To predict the weight of the samples of the *X2.csv*, we are going to give as prediction the score of our best model, to which we add 15% of the current value, rounded up. The logic is that on one side, the model will be trained on the whole dataset instead of only 90%, which is supposed to raise the accuracy. But on the other hand we have unseen data, still containing outliers, which will probably cause a relatively big drop in the regression score. We thus compute $1.15 * 0.1015 = 0.1167$, which rounded up gives 0.12.

6 Conclusion

In conclusion we standardized the data, then tested different preprocessing/data augmentation steps: feature selection, outliers removal, sampling weights, and oversampling. Only the two first techniques were used. We didn't reach good results with oversampling, we are however lucid in our own abilities and realize that we probably did several mistakes when implementing this technique. We then evaluated different models: a linear regressor (LR), a multilayer perceptron regressor (MLP), a K-Nearest Neighbors regressor (KNN), and a random forest regressor (RF). We selected the LR model for which a regression score of 0.1015 was reached.

We learned a lot of practical and theoretical knowledge about machine learning models, preprocessing, and particularly about data augmentation and class distribution with oversampling, undersampling, etc. We also gained

valuable generic insights such as "visualize the data, familiarize yourself with the data, think about what you do and why you do it".

There are several ways in which we could improve this project. One very simple thing would be to test more models (there are a lot of models available on sklearn alone). Another would be to have a second pass at preprocessing and data augmentation, trying to understand things we did wrong and correct those mistakes.

This concludes our report, thank you for reading through it, we are eager to hear your feedback.