Castiglione Thomas
Sciper N° 279053
december 2020

# BIO-322 MINI-PROJECT A REPORT

## Introduction

The goal of this project is to create a model to predict at best the pleasantness of the smell of a chemical compound based on some of its known chemical features. We will first visualize the data, then try some linear methods and finally some non-linear ones.
Looking at the data, we quickly see that we are in the n < p situation, meaning we will have to make choices to avoid overfitting. We can also see that the "Intensity" predictor is a character, so we perform one-hot coding to have a more suitable format of the data and we get rid of the predictor "SWEETORSOUR" since it is not on the final test set. We also get rid of the zero variance predictors, since they will not have any impact on the models. We perform a pairwise correlation check, looking for highly correlated predictors using the 'corrplot' library to visualize it. We can spot (on the randomly plotted predictors) some highly correlated predictors, so we remove one of each pair having a correlation > 0.99. We see later that the removal of these predictors does not have a significant impact on the test error. After this, we find ourselves with 1853 remaining predictors, a lot less than the 4871 from the raw data, but we still are in the n < p situation. To check the precision of our models, we separate our data between a training and a test set.

## Results

### *Linear methods*

We will first perform a very wrong linear regression using all predictors to confirm that the n < p situation leads to huge overfitting, which is the case. Now we need to choose and try methods countering this effect. We first choose to perform cross-validated subset selection, using forward selection (best subset selection would be computationally too expensive). We test our model on the test set and get a test RMSE of approximately 23.25 which is not bad for a linear method. An interesting observation is that the best model found uses only 3 predictors, and using only these we can say that it is surprisingly accurate, confirming that having a big amount of predictors in the raw data does not mean that the data depends on a lot of them. We know that the Lasso regularization with cross-validated chosen lambda parameter is a very efficient method (in accuracy and computational cost), so we perform it. And effectively, it is a lot faster and accurate than the cross-validated subset selection we made before, with a test RMSE of 21.60, making it our best linear model. But, when submitting this model to Kaggle we get an RMSE of 22.8 on the final test set. This may be caused by the noise in the final test set and our test set because they both are relatively short data sets. We may also have overfitted our own test set here. We also confirm that dropping the highly correlated parameters almost does not impact the test error.

### *Non-linear methods*

For non-linear methods, we directly tried artificial neural networks (NN) using the library "keras". This choice is motivated by the fact that neural networks may fit accurately a lot of different data distribution type, even if some other non-linear methods can be better in some cases. We then need to find the best tunning parameters. We decide to use neurons with relu activation function since it seems to work well in regression tasks. The input shape of the neural network is 1853 (number of conserved predictors), its output is a single linearly activated neuron (prediction). Our method for choosing the neural network depth, the number of neurons per layer, maybe some regularization parameters is the following: We start with a very simple network (let say 1 hidden layer with 8 neurons)

and we try some complexification little by little, choosing which direction is best each time by training the network with different seeds and looking at the validation or test error. When we see that we start to overfit, we choose to go back in complexity (like decrease the number of neurons in a layer or decrease the number of layers) and try some other directions of complexification. We also try to add some dropout layers and to regularize certain layers when we achieved what we think is the most accurate network we can get without overfitting and without any regularization. We also tried to add kernel regularizers and bias regularizers but the results were not that good. After hours of tunning these hyper-parameters, the best network we found has 6 hidden layers with respectively 60,80,40,10,10 and 10 relu-activated neurons, with a dropout layer with factor 0.1 between the 2nd and 3rd hidden layers. We chose the epochs parameters to be 100 because above this we did not see any improvement behind it. We also tried callbacks with a patience of 20 steps, to make sure we are starting to overfit before stopping the training. The average test RMSE we obtain (from different seeds) is 21.62 on our test data and 21.33 on the Kaggle test set. This is the best result we will get through this project, so it is our best model. We note that scaling the predictors will not have a large impact on the accuracy of the model here.
Since there was no huge improvement between linear regression and artificial neural networks (on our own test set), we thought that there could be 2 possibilities now: first, this data is fitted almost as good by linear regression and artificial neuron networks, and we got close than the limit RMSE coming from the noise of the response. Or, we still did not apply a method that suits this data best. Actually, we cannot verify any of these assumptions, so we decided to try gradient boosting and random trees using libraries “xgboost” and “randomForest”.
To tune gradient boosting hyper-parameters, we computed models with different values of parameters “nrounds” and “max_depth”, and computed the training and test. Then, we took the values corresponding to the lower test RMSE we got, and the test RMSE on the best hyper-parameters we found is 22.63, which is not better than what we got. We see from the importance plot from xgboost that there is a relatively big difference (factor 2) between the 2 most important predictors and the other most important ones. This could partially explain why Lasso regularization performs best with only 3 predictors.
We also ran random forests on different seeds to get an average test RMSE, which is around 22.65, almost the same as gradient boosting.

## Conclusion

In conclusion, we got our best model with an artificial neural network. It has 6 hidden-layers of relu-activated neurons disposed as: 60, 80, dropout of factor 0.1, 40, 10, 10, 10, and one output neuron with linear activation function.
From the Kaggle leaderboard, we can see that some models get RMSEs lower than 20, which is a big step forward from our 21.33. This really gives us an overview of the capacity of machine learning and so the desire to learn more about machine learning methods.