Machine Learning for Level Truncation in Open String Field Theory
Notes

Harold Erbin and Riccardo Finotello

Dipartimento di Fisica, Università di Torino
and I.N.F.N. - sezione di Torino
Via P. Giuria 1, I-10125 Torino, Italy

29th June 2020

1 Synopsis

In the framework of bosonic Open String Field Theory (OSFT) we consider the solutions at different radii of several observables and at different mass level truncations. We then use machine learning (ML) techniques to extract the value of the observable at infinite mass level truncation. More details can be found looking at the analysis at this URL (only the analysis is accessible, while the original database is not public).

2 Preliminary Operations

2.1 Description

The dataset is composed of 46 different solutions at different radii. Each of them is then composed of lists of several observables of different lengths (varying from 15 to 21 entries each, as shown in Figure 2.1). Every observable is characterised by its conformal weight (the `weight` column in the dataset), its kind of "oscillations" (`type` variable, categorical and ordered), its initialisation point (`init`) and the truncation levels (from level 2 to level 18). The objective of the analysis is the prediction of the extrapolation for the level-∞ truncation hosted in the `exp` variable of the dataset and which take 3 possible integer values in the range $[-1, 1]$.

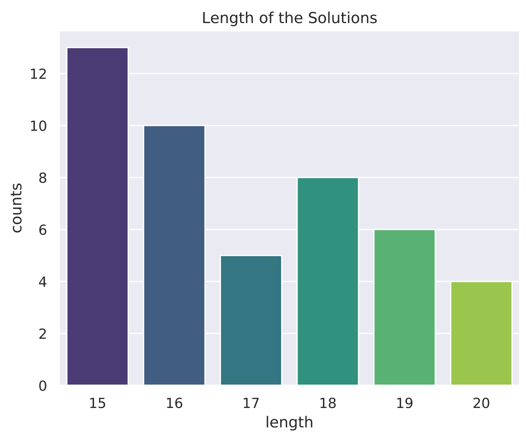

Figure 2.1: Length of the solutions in the untidied dataset.

2.2 Input Preparation

For the analysis we extract each entry of the lists and put it in separate entry of the dataset: we first artificially insert a new variable labelling the solution with a number from 0 to 45 and then flatten the dataset over the rows. This way all columns hold only numerical variables which can be used for different steps of the analysis. The tidy dataset is composed of 778 samples over 22 columns (including the newly created `solutions` variable).

We then look for duplicates inside the tidy dataset: rows which present exactly the entries over all the columns, and exclude them from the analysis (we keep only the first occurrence). The final version of the dataset holds 732 samples and can be used for the exploratory data analysis (EDA).

3 Exploratory Data Analysis

During the EDA we study properties of the entries of the tidy dataset. In particular we focus on outlier detection, distribution of the features, correlations between variables, principal components and clustering. We also anticipate that we will not use the `solutions` and `init` variables in the regression analysis and as such we do not include them in the EDA as well: they only serve as labels for the identification of the entries.

3.1 Outliers Detection

We perform outliers detection using the definition of the *interquartile* range, typical in statistical analysis. This is defined as the interval between the 25th and 75th percentile of the sample distribution, and outlying samples are entries greater or lower than 1.5 times such interval with respect to the sample mean.

We apply this procedure for each column in the dataset and compute the average quantity of outlying samples for each variable: the truncation levels hold between 17% and 27% of outliers in their distributions, while `weight` holds only 6% of outliers with respect to its distribution (the `type` feature is categorical and as such there is no point in studying the outlying samples for it). The distribution of the samples is summarised in Figure 3.1: we digitised each variable in at most 10 bins representing left-closed intervals (i.e. $[a, b)$) between the values reported on the x-axis. As we can see the distributions present a lot

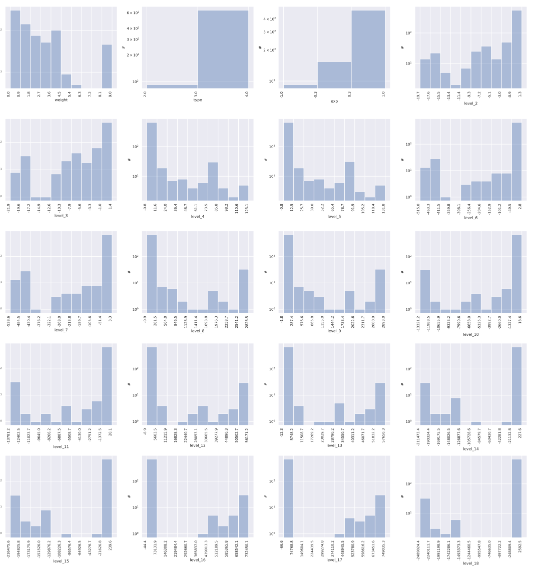

Figure 3.1: Distribution of the variables in the dataset.

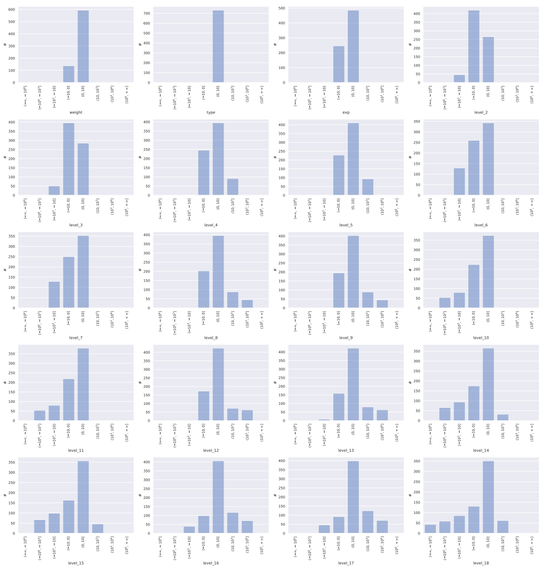

Figure 3.3: Distribution of the variables categorised by order of magnitude.

of outlying samples which may lead to a "artificially too small" variance on the determination of the coefficients of the regression. The outliers are however mainly present in the distribution of the variables for which **weight** ≥ 1.5. In fact if we consider **weight** < 1.5 the variables are more

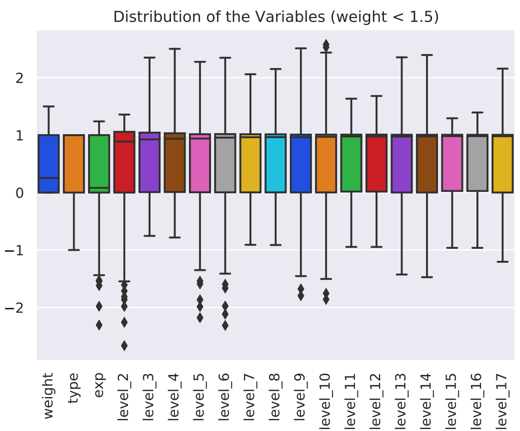

Figure 3.2: Distribution of the variables for **weight** < 1.5. The bars show the quartile of the distribution (the marked horizontal line inside the bars is the median, or second quantile), while the "wiskers" extend to cover the interquartile range. Isolated points are the remaining outliers of the distribution.

reasonably distributed, and might be easier to train in a following regression analysis, as we show in Figure 3.2.

We can also show a plot similar to Figure 3.1 but categorised by order of magnitude to better show the distribution of large samples. As we see in Figure 3.3, even though outliers are present, most of the samples are contained in the intervals $(-10, 10]$ which at least ensures that we are dealing with centered distributions, though unbalanced.

3.2 Correlations

Following the outliers detection analysis, we first notice that for different ranges of the **weight** variable, only certain types of oscillations are present. In fact, in Table 3.1

		weight	
range	*type*	*mean*	*variance*
weight ≥ 1.5	4	4.03	5.13
weight < 1.5	2	0	0
	4	0.58	0.21

Table 3.1: Type of oscillations per weight range.

we show that for higher weights there is only one type of oscillations in the dataset, while for lower weights the type 2 oscillation implies **weight** $= 0$ identically.

This in turns may also be reflected in the correlation between the variables. This is defined for each possible couple of variables as the ration between their covariance and the product of their separate standard deviations. In Figure 3.4 we show the different correlation factors in a graphical representation. As expected, the truncation levels are strongly correlated among themselves and with the **weight** variable. Though milder, there is also a good correlation of the variables with the labels we intend to predict (**exp**), while the **type** variable seems to be completely uncorrelated (this may however be due to the fact of being categorical: a linear regression might be more suitable to do statistical inference on it). Another notable remark is the different behaviour of the correlations between consecutive layers: the large anticorrelations seem to be entirely due to the higher weights, while **weight** < 1.5 seems to imply a correlation only between adjacent couples of levels (e.g. 2-3, 4-5, 6-7, etc.) and their replicas at distance 3 (e.g. 4-5 with 8-9, 12-13

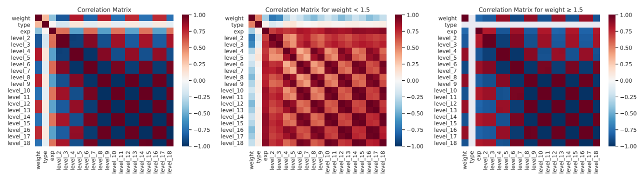

Figure 3.4: Correlation matrix of the variables.

and 16-17, 6-7 with 10-11, 14-15 and 18, etc.). This behaviour may point to several possible interpretations. In particular we see that data is oscillating: going towards larger truncation levels data seem to alternate a behaviour similar to a periodic function (sine or cosine in the particular case) with maxima and minima alternating at constant intervals. This may become clear when considering the definition of the correlation factor of two variables X_1 and X_2 given as the ratio between the covariance $\sigma(X_1, X_2) = (X_1 - \bar{X}_1) \cdot (X_2 - \bar{X}_2)$, where $\bar{X}_1$ and $\bar{X}_2$ are the mean of the respective variables, and the product of the separate standard deviations $\sigma(X_1)\sigma(X_2)$). However we also see that this behaviour is particularly present when considering the full dataset (or just `weight` ≥ 1.5) while if we introduce a cut at `weight` < 1.5 the correlation is definitely less marked, though present.

3.3 Principal Components Analysis

We perform the Principal Components Analysis (PCA) of the truncation levels to study their properties and their distribution: it encodes the linear variance of the samples by projecting the distribution onto a new system of coordinates trying to maximise such variance in each new axis. This approach may be interesting both as exploratory analysis and for further directions: we may want to generalise the algorithm to an arbitrary number of truncation levels (and we may want to skip some values). PCA would then give a solution to always have input of the same size while keeping as much information as possible.

We first study all its components using the Singular Value Decomposition (SVD) of the matrix holding samples over the rows and the truncation levels over the columns (it is a rectangular matrix and as such cannot be strictly diagonalised to perform spectral analysis). In Figure 3.5 we show the variance explained by each principal component of the matrix (i.e. the fraction of variance of the total set retained by considering only the selected component).

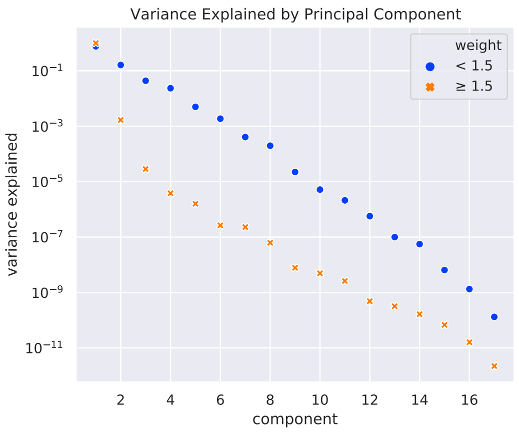

Figure 3.5: Variance explained by the principal components of the truncation levels (log scale).

The analysis was performed splitting the dataset over ranges of the `weight` variable and shows that for lower weights the first two principal components account for more than 90% of the total variance of the values, while for larger weights almost 100% of the variance is explained by the first component. This reflects the distribution of the variables in the dataset: larger weights contain a very large amount of outliers and have larger variance with respect to lower weights, thus it might be enough to project the values of the truncation levels over the line which contains most of the deviation to reproduce a meaningful distribution.

3.4 K-Means Clustering

Finally we perform an unsupervised clustering analysis. The main idea is to infer a structure in the data which may be able to "automatically" reproduce the labels (i.e. the `exp` column) without regression. In other words we study the distribution of the truncation levels and fit it in 3 clusters representing the 3 integer values of the labels. In the ideal scenario there should be a 1:1 relation between the labels of the cluster centroids and the labels in the `exp` variable. Unfortunately the cluster analysis of the truncation levels over the entire dataset highlighted no particular structure in the data and turned out inconclusive.

We then performed the same analysis splitting the dataset in different ranges of the `weight` variable, standardising the data for `weight` < 1.5 and robustly scaling (i.e. scaling according the interquartile range) samples for which `weight` ≥ 1.5. Based on the results of Section 3.3, in Figure 3.6 we used the principal components of the truncation levels to plot the distribution of the clusters and the `exp` labels. As we can see, recognising a structure is challenging when `weight` ≥ 1.5 and in fact different `exp` (or "extrapolation") labels generally belong to different clusters. However the case `weight` < 1.5 seems to be more accurate and shows that in general we can recognise a structure in the data. In Table 3.2 we recognise that there is a distinguishable relation between the extrapolation column and the average cluster label in the group when `weight` < 1.5, while in general there is a superposition of labels when `weight` ≥ 1.5 (it seems that there are only 2 recognisable clusters).

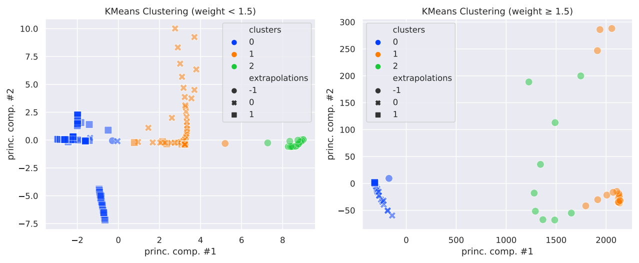

Figure 3.6: K-Means clusters and `exp` labels plotted using the principal components for visualisation purposes.

		weight	
clusters	*exp*	< 1.5	≥ 1.5
average label	-1	1.84	1.13
	0	0.92	0.00
	1	0.02	0.00

Table 3.2: Average cluster label per weight range.

4 Regression Analysis

In what follows we decided to drop the first type of solution from the dataset (i.e. `solutions` = 0) since it looks "too perfect" for the analysis and might spoil the final results (also, removing a restricted number of samples does not change the underlying distribution, as shown earlier).

4.1 Test and Validation Sets

The splits for training and test sets have been chosen in order to keep samples coming from the same `solutions` inside the same set in order to maintain a good balance in the distribution of the variables. In other words, we choose the samples in the test and training sets by first subsampling the unique values of the `solutions` column, then we split them into separate sets, and ultimately we assign the corresponding samples to the splits.

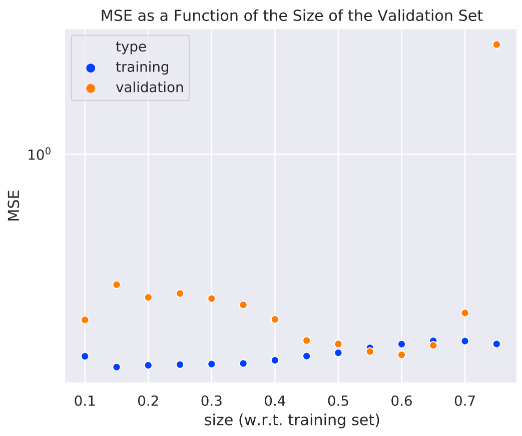

Figure 4.1: MSE in training and validation for several sizes of the validation set.

We first decide to subsample around 10% of the solutions for the test set and keep the remaining 90% for training and validation. Notice that we do not use cross-validation in this case since the number of samples is very restricted and we still want to keep samples coming from the same `solutions` in the same split.

We chose the size of the validation set to be around 10% of the total training set as it is typical for datasets of this dimension where most of the samples should be assigned to training.

Another motivation for the choice is to try to minimise the error (the Mean Squared Error, MSE) made when fitting a linear regression on decreasing size of the set effectively used for training, after removing the validation split: this may clearly not work for anything different from a linear model, but it is however a good starting point. In Figure 4.1 we show training and validation MSE for different sizes of the validation set (with respect to the total training set). In order to keep a reasonable amount of samples in the training split and to contain the validation MSE as much as possible, we finally take the already announced 10% of the unique `solutions` in the validation set. In Table 4.1 we give a schematic summary of the final choice for training, validation and test sets.

	total dataset	
	samples	*fraction*
training	579	81%
validation	61	8%
test	78	11%

Table 4.1: Summary of the train/validation/test splits.

4.2 Preliminary Linear Regression Analysis

Before moving to the ML analysis, we first perform a preliminary study using linear regression to infer properties of the variables which may play a role in the ML algorithms. From now on we shall consider the training data as robustly scaled against the outliers (i.e. we pre-process using `sklearn.RobustScaler`).

We first fit the linear regression algorithm on the train set and then compute the predictions on the three splits to study the values of the coefficients, their associated standard error, t-statistics and probabilities of being different from zero. For this and the ML analysis we do not fit the intercept of the models since the variables cannot vanish altogether: the intercept has therefore no interpretation in discovering a structure in the values or performing regression.

The study revealed that when considering the full dataset, all the variables should be considered for regression as the coefficients are non vanishing (with 95% confidence), mostly due to the very large variance of the data which allows only very precise (i.e. very small standard errors) determinations of the coefficients. The same study performed on the usual separate ranges of `weight` showed that when considering `weight` ≥ 1.5 the variable `type` is not necessary to the fit and we cannot reject the hypothesis of a vanishing coefficient for such feature: this in fact reflect one of the properties discovered during the EDA and summarised in Table 3.1 where we showed that for `weight` ≥ 1.5 only one `type` is present in the dataset and does not influence the final predictions. We will however keep such feature inside the training variables since a further study on the full dataset without the categorical variable showed that predictions are in general worse than in the previous cases (i.e. the `type` variable helps when `weight` < 1.5).

5 Machine Learning Analysis

In this section we briefly summarise the results of the ML analysis performed on the data. We will explore several diverse algorithms including regularised versions of the linear regression (LR, from now on), support vector machines (SVM), decision trees (random forests, RF, and gradient boosted decision trees, GBDT), and artificial neural networks (ANN). For each algorithm we fit the data on training set and evaluate it on the validation set to perform the optimisation of the *hyperparameters* using Bayes statistics (see [1]–[4] for the modules used in the analysis: `scikit-learn` for linear models and SVM, `lightgbm` for trees and `tensorflow` for the ANN). The goal is to minimise the objective function, that is the validation MSE, without knowing its analytic form, by only inferring the probability of obtaining a better result with a different choice of hyperparameters.

5.1 Regression Analysis

In Table 5.1 we summarise the results obtained from prediction on the test set for the algorithms used: in particular the first four lines refer to linear models (LR is the plain regression, EN implements both ℓ_1 and ℓ_2 linear regularisation (i.e. it introduces in the linear least square minimisation terms proportional to the norm and the squared norm of the parameters of the fit, weighted by two separate terms), Lasso and Ridge separately implement ℓ_1 and ℓ_2 regularisation, respectively); l-SVR is the linear implementation of the SVM models, while r-SVR uses a Gaussian *kernel trick*; RF and GBDT are different ensemble algorithms based on decision trees (namely random forests and gradient boosted trees); ANN represents the neural network architecture (no hidden layers, only one input layer with 30 units, batch normalisation layer and a dropout layer with 0.1 drop rate leading to a network of 691 trainable parameters and 60 fixed parameters).

	MSE	MSE 95% CI	MAE	R2
LR	0.13	$[0.07, 0.20]$	0.27	0.74
EN	0.19	$[0.12, 0.27]$	0.33	0.62
Lasso	0.20	$[0.13, 0.28]$	0.34	0.60
Ridge	0.13	$[0.07, 0.20]$	0.27	0.74
l-SVR	8×10^3	$[-3, 20] \times 10^3$	20	-1.6×10^4
r-SVR	0.030	—	0.08	0.95
RF	0.013	—	0.05	0.97
GBDT	0.002	—	0.03	1.00
ANN	0.003	—	0.04	0.99

Table 5.1: Summary of the predictions on the test set for the different algorithms. We present the MSE, the Mean Absolute Error (MAE) and the R^2 score. The confidence intervals of the MSE are present only for linear models since they rely on statistical assumptions on the distribution of the residuals (they are normally distributed).

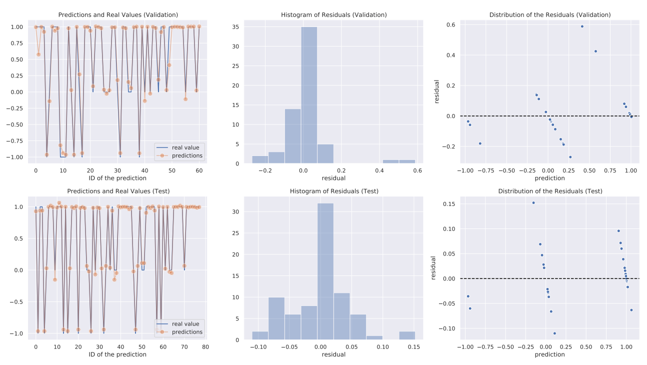

Figure 5.1: GBDT validation and test sets residuals. When showing the predicted values compared to real values the markers show the position of the predictions and the line connecting then should help in following the sequence. True values are located at the angled points of the blue lines.

As we can see the GBDT are able to deliver the best results both in terms of MSE and R2 score followed by the implementation of the shallow network (trained over 5000 epochs: training is fairly rapid but trees are by far faster). On the other hand ℓ_1 regularisation (both alone and in combination with ℓ_2) fails to approximate well the prediction labels. In the same fashion the l-SVR fails completely.

In Figure 5.1 we show the validation and test set distribution of the predictions and residuals. As we can see the algorithm correctly approximates the `exp` labels with residual contained in a small interval. There is however a general tendency to underestimate the `exp` $= -1$ label and to overestimate `exp` $= 1$: in linear model this would have signalled an incomplete model since residuals seem to be correlated with predictions, but in the case of decision trees it does not pose an issue.

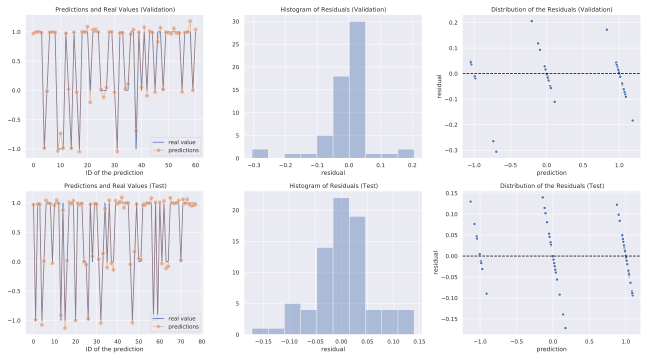

Figure 5.2: ANN validation and test sets residuals. You can refer to the previous plot for an explanation of the graphical signs.

In Figure 5.2 we show the same plot generated from the ANN predictions. As for the GBDT, the ANN is correctly trying to predict the `label` with very small residuals. Differently from the GBDT, residuals are more randomly distributed for each prediction and no pattern is recognisable.

	RF	**GBDT**
no. leaves	48	25
max depth	300	6
no. estimators	25	3345
subsample	0.85	0.99
colsample by tree	0.7	1.0
min child weight	0.01	0.1
ℓ_1 reg.	0.16	2.1
ℓ_2 reg.	0.20	690
learning rate	—	0.05

Table 5.2: Hyperparameters choices for RF and GBDT.

In Table 5.2 we show a summary of the hyperparameters used for training RF and GBDT: as expected the optimisation process chose a small number of fully grown trees in the first case, while it led to a large number of boosting rounds of shallow trees in the latter.

5.2 Feature Explanation

As a last step in the analysis, we use the trained trees (for simplicity) and study their properties to possibly show how each training feature plays inside the algorithm and how the final prediction is determined. In this part of the analysis we study the variable ranking (i.e. the importance of the features) provided by the binary structure of the trees and the Shapley values [5], [6] of each feature. The importance of the features is a key property of the decision trees and visually summarises the impact that each feature has on the final prediction: variables with higher importance can be found in the first branches of the trees because they are responsible for the main choices of the algorithm, while more negligible features provide the necessary refinement. The Shapley values are somewhat related and derive from a game theoretic approach to the decision trees: these values encode how a certain feature is influencing the final result, whether by dragging its value with respect to average of the predictions or by pushing it beyond it. In other words, feature importances show how much (as a percentage of the final prediction) a variable is relevant for the algorithm and the Shapley values show if there are variables which tend to drive the results towards a particular value (by computing the interaction between Shapley values we can also try to recognise interdependencies inside the trees which may be worth analysing further since the EDA could not have possibly picked them up, since they are a product of the training process).

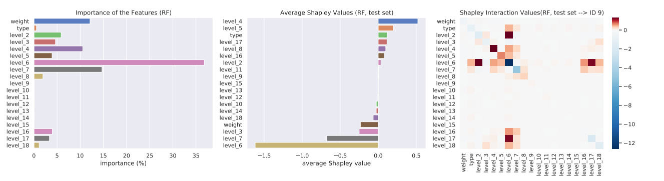

Figure 5.3: Variable ranking and Shapley values (computed on the test set) of RF.

In Figure 5.3 we show the variable ranking and the Shapley values for the RF algorithm. The first plot sow that RF rely on a particular truncation level (level-6 to be specific) to produce the predictions, while other features contribute in a less relevant manner. The average Shapley values (second plot) show that most features give a comparable contribution to the prediction, exception made for the mass truncation at level-6 and level-7 which seem to drive the final result in a more direct way (according to Figure 3.1 they are among the highly unbalanced values, which may be a reason of such behaviour). Finally the *interacting Shapley values* (taken for a random sample in the test set) in the third plot show that the whole set of variables is mostly non interacting except very sporadic contributions between distant levels.

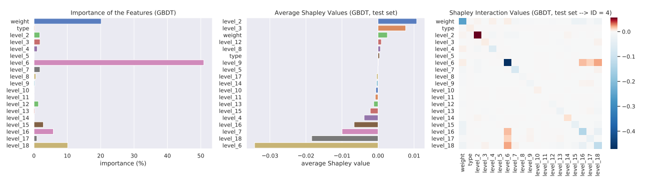

Figure 5.4: Variable ranking and Shapley values (computed on the test set) of GBDT.

In Figure 5.4 we finally show the same plot for the GBDT. Differently from the RF, GBDT seem to rely almost entirely on the 6th truncation level and `weight` discrimination of the samples.

In both case the algorithms seem to ignore almost completely the categorical variable `type` (possibly it could have been removed but the corresponding Shapley value shows that its relevance is so marginal that the result would not have changed).

We notice that the scale of the Shapley values is very different between RF and GBDT: this is due to the fact that in this case the boosting procedure may be more robust against the large variability of the dataset, showing that in the GBDT case the trees are more balanced. Ultimately this is also a consequence of the different order of magnitude of the MSE associated to RF and GBDT, the latter being an entire order more precise than the former.

6 Conclusions

We showed that the dataset has an heterogeneous and sparse distribution when not accounting for the range of the `weight` variable. In particular, in the case of low weight the distribution is contained and shows an underlying distribution which unsupervised learning can capture and connect to the prediction labels. Discriminating over the `weight` variable also revealed that some choices of the `type` variable are somewhat forced (see Table 3.1), leading to consequences in the analysis of linear regression in Section 4.2. The ML analysis revealed that the best algorithms for the prediction of the labels are GBDT and ANN which reproduce correctly and without overfitting (we did not show explicitly the numbers, but the validation MSE are comparable to the test results shown in Table 5.1). We finally showed how decision trees treat each variable during predictions on the test set and revealed that features rarely influence each other. In general the predictions of the trees and the ANN seem to be reliable and consistent, suggesting that indeed the prediction of the `exp` labels is an achievable task.

References

[1] F. Pedregosa, G. Varoquaux, A. Gramfort *et al.*, 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[2] T. Head, MechCoder, G. Louppe *et al.*, *Scikit-optimize/scikit-optimize: V0.5.2*, version v0.5.2, Mar. 2018. DOI: 10.5281/zenodo.1207017. [Online]. Available: https://doi.org/10.5281/zenodo.1207017.

[3] G. Ke, Q. Meng, T. Finley *et al.*, 'Lightgbm: A highly efficient gradient boosting decision tree', in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio *et al.*, Eds., Curran Associates, Inc., 2017, pp. 3146–3154. [Online]. Available: http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf.

[4] Martín Abadi, Ashish Agarwal, Paul Barham *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[5] S. M. Lundberg and S.-I. Lee, ‘A unified approach to interpreting model predictions’, in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio *et al.*, Eds., Curran Associates, Inc., 2017, pp. 4765–4774. [Online]. Available: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[6] S. M. Lundberg, G. Erion, H. Chen *et al.*, ‘From local explanations to global understanding with explainable ai for trees’, *Nature Machine Intelligence*, vol. 2, no. 1, pp. 2522–5839, 2020.