Decision Tree Regressor and Random Forest Regressor

All the machine learning (ML) algorithms used in this work were applied as implemented in the *sklearn* Python library.[i] The multivariable linear regression (MLR) simply corresponds to the ordinary least squares linear regression but as a function of two or more variables. The decision tree regressor (DTR) is a supervised learning algorithm that has, as a goal, to build a model predicting the value of a target variable, in our case the E_{ads} or E_{abs}, through the learning of a set of binary rules derived from the data features, here the descriptors and surface features listed in the manuscript. For this DTR builds a model in the form of a tree structure featuring branches, nodes, and leaves. The order of the questions —known as decision nodes, which gives the name to the method— as well as their content is automatically determined and optimized by the algorithm, which looks for the homogeneity of y values in a found subset, and for that, uses the standard deviation of y values as an optimization criterion. During the training, the model learns any relationships between the data and the target variable, defining the best questions as well as their order to make the most accurate estimates possible. When predicting the dependent variable value of a new data point, this point is run completely through the entire tree branches by answering the node questions —posed as logic steps— until reaching the final answer leaf, with the target variable. Indeed, this variable value is just the average value of all points satisfying the same logic questions as the dependent point.

The complexity of the above explanation calls for an illustrative example so as to understand the basics of the procedure. Let us define a dependent variable target, y —which could be, *e.g.*, E_{ads}—, which depends on two features, x_1 and x_2 —*e.g.*, two descriptors, such as ε_d and γ. Figure 1 displays a scatter plot of x_2 *vs.* x_1, and one could well imagine a three-dimensional plot, where y values would define the points heights in an orthogonal axis to x_1 and x_2. In it, dotted lines would delimit regions defined by the nodes, *e.g.*, a first one asking whether x_1 values is larger than 50, and second ones asking whether x_2 value is larger than 4 for x_1 values smaller than 50, or x_2 value being larger than 12 when x_1 is larger than 50. That leads to final four leaves, with different average y values, $\bar{y}$, see Figure 2. For instance, when this is done for a training set, a test set value with $x_1 = 25$ and $x_2 = 12$ would yield an expected $\bar{y}$ value of 5.1. Notice that this is done for a very simplified, two-dimensional case, with y values depending on two variables, but is easily applied for the dependence on a higher number of variables.

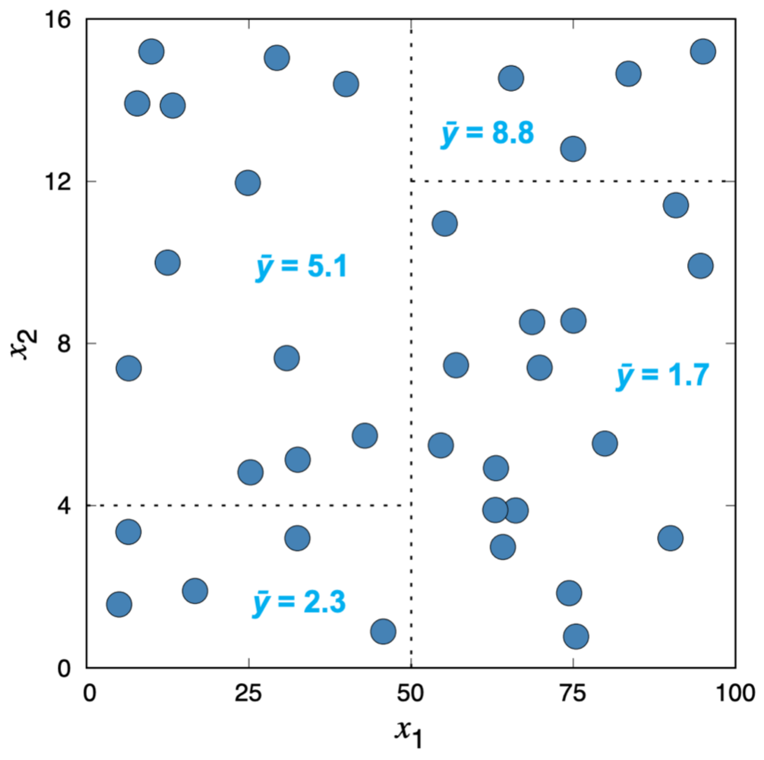

Figure 1. Exemplary scatter plot of values, shown as blue circles, of x_2 *vs.* x_1 variables, where black dotted lines represent the variable splitting decisions learned by the model. In light blue, the $\bar{y}$ average values for all those points belonging in each of the resulting sections inside the plot.

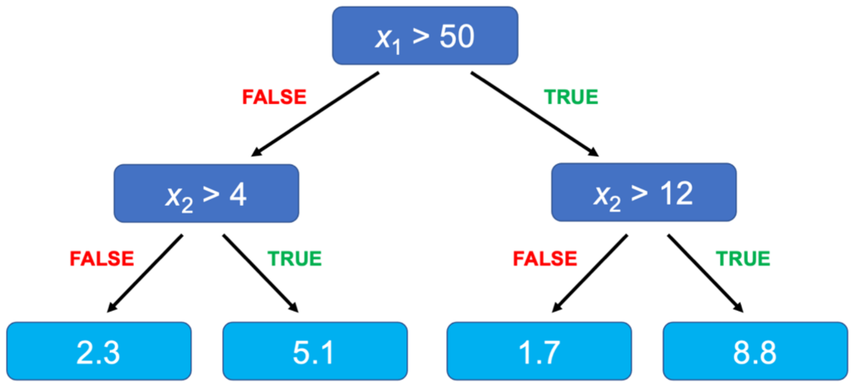

Figure 2. Exemplary decision tree from data shown in Figure 1.

Notice that, even if DTR is relatively easy to interpret, and requires little data preparation, the resulting tree is sensitive to the employed data, *i.e.* small variations on the data may be translated into a completely different built tree. To overcome this drawback, an ensemble of trees, *a.k.a.* a forest, can be used. This is indeed the basics of the random forest regression (RFR), where an ensemble of decision trees is grown. Each tree is assembled from a sample randomly drawn from the training set. For each tree, when splitting each node, the best split can be found either from all the input features or for a random subset of features, *a.k.a. max_features*. These two sources of randomness help decreasing the variance of the estimator, since individual decision trees usually exhibit high variance and tend to overfit. The background idea is that, when a prediction is cast on a data set, *e.g.* the above commented x_1 and x_2 values, different $\bar{y}_i$ values are obtained for $i = 1 - N$, where N is the number of trees of the forest, a.k.a. *n_estimators*. Thus, the expected value of $\bar{y}$ is simply the average over the expected $\bar{y}_i$ values on the different N trees. By doing so, the predictive accuracy of the trees is narrowed, diluting possible extreme $\bar{y}$ forecasts. Notice that RFR becomes DTR for $i = 1$, and that the more the trees, the better the accuracy is, yet computationally more expensive. Aside, notice that accuracy decays with N, but results do not normally improve beyond a critical number of trees.

An appealing aspect of both DTR and RFR is the fact that the relative rank of a feature used as a decision node in a tree can be used to assess the relative importance of that feature with respect to the target variable predictions. For instance, features used at the top of the tree have a larger impact, since they affect the final prediction decision of a larger fraction of samples. Hence, the relative importance of the features can be estimated as the expected fraction of samples they contribute to.

K-means Clustering

The k-means (KM) algorithm has been used as implemented in the *sklearn* python library.[i] KM clusters data by splitting samples in *n* groups (clusters) of equal variance, where *n* is defined beforehand. The algorithm automatically finds *n* cluster centres which minimize the within-cluster sum-of-squares, also called inertia, see Eq. 1, in order to avoid compensation by negative displacement vectors from the centre, and as a way of preferentially gather those data points close to the cluster centre, and, at the same time, bias those located farther away.

$$\sum_{i=0}^{n} \min_{\mu_j \in C}\left(\left\|x_i - \mu_j\right\|\right)^2 \qquad (1),$$

where μ_j is the samples mean, also called cluster centroid, in the *C* disjoint clusters.

Inertia can be understood as a quantitative measure of how coherent clusters are, yet has some drawbacks. It assumes that clusters are isotropic and convex, and so, performs poorly with elongated clusters. Moreover, note that inertia is a non-normalized metric; *i.e.* one can just state that lower values are better, and zero optimal. In that regard, it does not perform well in high-dimensional spaces, where Euclidean distances tend to become inflated. Running a dimensionality reduction algorithm prior to the k-means analysis can help alleviating this problem.

However, that is not the case in the present study, and so this method is suited for the employed data, displayed two-dimensionally. Note that there exist other methodologies to find and define groups of similar behaviour, such as the subgroup discovery (SGD),[ii] where a target variable is expressed as a function of a series of description features, which are Monte Carlo optimized to define regions of subgroups, as done *e.g.* in the past in the categorization of binary structures to rocksalt or zinc blende structures as a function of the atomic radii of valence *s* and *p* orbitals.[iii] However, SGD defines fringe conditions from which one expects one or another behaviour, while here KM defines centroids whose close data implies a similar behaviour.

Focusing on KM, notice that when *n* is exactly the number of data points, the minimum inertia of zero is achieved. Therefore, one needs to define a criterion to restrict to a small and useful *n* number. This has been done using the elbow method,[iv] which consists in plotting the inertia as a function of *n* and picking the elbow of the curve as the suited number of clusters to use.

References

[i] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. Scikit-Learn: machine learning in Python. *J. Mach. Learn Res.*, **12**, 2825-2830 (2011).

[ii] Duivesteijn, W., Feelders, A. J. & Knobbe, A. Exceptional model mining. *Data Min. Knowl. Discovery*, **30**, 47 (2016).

[iii] Goldsmith, B. R., Boley, M., Vreeken, J., Scheffler, M. & Ghiringhelli, L. M. Uncovering structure-property relationships of materials by subgroup discovery. *New. J. Phys.*, **19**, 013031 (2017).

[iv] Chaker, Z., Chervy, P., Boulard, Y., Bressanelli, S., Retailleau, P., Paternostre, M. & Charpentier, T. Systematic method for the exploration, representation, and classification of the diphenylalanine solvatomorphic space. *J. Phys. Chem. B*, **125**, 9454-9466 (2021).