Machine Learning Course
Project Report
“Predicting Molecular Properties”
(Kaggle competition)

Fedor Ivachev, ID 2019280373
Sami Emre Erdogan, ID 2019280513
Andrei Glinskii, ID 2019280807
Tsinghua University, Beijing, China

Contents

1. Introduction

1.1. Description of the competition

This challenge aims to predict interactions between atoms. Imaging technologies like MRI allows us to see and understand the molecular composition of tissues. Nuclear Magnetic Resonance (NMR) is a closely related technology which uses the same principles to understand the structure and dynamics of proteins and molecules.

Researchers around the world conduct NMR experiments to further understanding of the structure and dynamics of molecules, across areas like environmental science, pharmaceutical science, and materials science.

In this competition the task is to develop an algorithm that can predict the magnetic interaction between two atoms in a molecule (i.e., the **scalar coupling constant**).

Using NMR to gain insight into a molecule's structure and dynamics depends on the ability to accurately predict so-called "scalar couplings". These are effectively the magnetic interactions between a pair of atoms. The strength of this magnetic interaction depends on intervening electrons and chemical bonds that make up a molecule's three-dimensional structure.

Using state-of-the-art methods from quantum mechanics, it is possible to accurately calculate scalar coupling constants given only a 3D molecular structure as input. However, these quantum mechanics calculations are extremely expensive (days or weeks per molecule), and therefore have limited applicability in day-to-day workflows.

A fast and reliable method to predict these interactions will allow medicinal chemists to gain structural insights faster and cheaper, enabling scientists to understand how the 3D chemical structure of a molecule affects its properties and behavior.

Ultimately, such tools will enable researchers to make progress in a range of important problems, like designing molecules to carry out specific cellular tasks, or designing better drug molecules to fight disease.

1.2. Data summary

Table 1-1. train.csv key summaries and the first row

id	molecule_name	atom_index_0	atom_index_1	type	scalar_coupling_constant
	85012 unique values			3JHC — 32% 2JHC — 24% Other (6) 43%	
0	dsgdb9nsd_000001	1	0	1JHC	84.8076

Table 1-2. test.csv key summaries and the first row

id	molecule_name	atom_index_0	atom_index_1	type
	45777 unique values			3JHC 32% 2JHC 24% Other (6) 43%
4659076	dsgdb9nsd_000004	2	0	2JHC

Table 1-3. structures.csv key summaries and the first row

molecule_name	atom_index	atom	x	y	z
130789 unique values		H 51% C 35% Other (3) 4%			
dsgdb9nsd_000001	0	C	-0.012698135900000001	1.0858041578	0.008000995799999999

Table 1-4. scalar_coupling_contributions.csv key summaries and the first row

molecule_name	atom_index_0	atom_index_1	type	fc	sd	pso	dso
85012 unique values			3JHC 32% 2JHC 24% Other (6) 43%				
dsgdb9nsd_000001	1	0	1JHC	83.0224	0.254579	1.25862	0.27201

Table 1-5. potential_energy.csv key summaries and the first row

molecule_name	potential_energy
130789 unique values	
dsgdb9nsd_000001	-40.5236795

Table 1-6. milliken_charges.csv key summaries and the first row

molecule_name	atom_index	mulliken_charge
130789 unique values		
dsgdb9nsd_000001	0	-0.5356890000000001

Table 1-7. magnetic_shielding_tensors.csv key summaries and the first row

molecule_name	atom_index	XX	YX	ZX	XY	YY	ZY	XZ	YZ	ZZ
130789 unique values										
dsgdb9nsd_000001	0	195.3147	0.0	-0.0001	0.0	195.3171	0.0007	-0.0001	0.0007	195.3169

Table 1-8. dipole_moments.csv key summaries and the first row

molecule_name	X	Y	Z
130789 unique values			
dsgdb9nsd_000001	0.0	0.0	0.0

dsgdb9nsd_000001.xyz (.xyz file example)

```
5
C -0.0126981359 1.0858041578 0.0080009958
H 0.0021504160 -0.0060313176 0.0019761204
H 1.0117308433 1.4637511618 0.0002765748
H -0.5408150690 1.4475266138 -0.8766437152
H -0.5238136345 1.4379326443 0.9063972942
```

2. Approaches for solving the task

2.1. Decision trees

Decision tree approach exploits the idea that the problem can be considered as geometrical rather that specific physical or chemical problem. We can assume that If there are two similar sets of atoms with the same amount of distances and the same types between them, we can assume that, the scalar coupling constant should be quite similar as well. Atoms which are closer to the pair atoms, with under consideration it will have a greater influence on the scalar coupling constant than the distant atoms. Thus, the problem can be dealt with tree-based algorithms — we need to find a model which can determine similar configurations with similar feature sets.

To get necessary representation of features for tree-based models we need to transform cartesian coordinates. Cartesian representation is not stable because each coupling pair is located in a different point in space and two similar coupling sets would have very different X,Y,Z coordinates.

We need to transform the features:

- Take each pair of atoms as two 1st core atoms
- Find the center between these atoms
- Find n-nearest to the center atoms
- Take the closest two to the center atoms as 3rd and 4th core atoms
- Calculate the distances from 4 core atoms to the rest of the atoms as well as to the core atoms.

Using this representation each atom position can be described by 4 distances from the core atoms. This representation is stable to rotation and translation and can be used for pattern-matching. We used LightGBM library for this approach.

2.2. Graph Neural networks

Many fundamental relationships between data in various fields of science and technology, such as molecular chemistry, molecular biology, pattern recognition, computer vision, and data mining, can be represented as graphs. Graph neural networks (GNN) have become increasingly popular in various fields, including the social networks, knowledge graphs, recommendation systems, and life science. The power of GNN in modeling dependencies between nodes in a graph makes it possible to make a breakthrough in the field of research related to graph analysis.

We created one more model, called graph neural network (GNN), as a second model for predicting molecule properties. Our model is based on the Chainer library. Chainer is an open source deep learning framework written purely in Python on top of Numpy and CuPy Python libraries. Main advantages of the Chainer are the "define-by-run" scheme and existence of Chainer Chemistry — a collection of tools to train and run neural networks for tasks in biology and chemistry using Chainer.

Chainer was the first library where define-by-run approach was implemented. The traditional network training procedure consists of two stages: Determining fixed relationships between mathematical operations (such as matrix multiplication and nonlinear activations) in the network, and then performing the actual training calculations. This is called the define-and-run approach, or static graph. Theano and TensorFlow are among the notable libraries that have adopted this approach. In contrast, in dynamic graph approach the network connection is not defined when the training has started. The network is defined in the learning process as the actual calculation is performed. One of the advantages of this approach is that it is intuitive and flexible. If the network has complex control flows, such as loops, the define-and-execution approach requires specially designed operations for such constructs. On the other hand, the define-by-run approach can use native programming language constructs such as if statements and for loops to describe such flows. This flexibility is particularly useful for implementing Graph Neural Networks. Define-by-run approach has gained popularity since the introduction of Chainer and is now implemented in many other frameworks, including PyTorch and TensorFlow. Chainer Chemistry tools allow to implement state-of-the-art deep learning neural network models for chemical molecules (NFP, GGNN, Weave, SchNet etc.) and preprocess datasets for molecules data.

Graph structure is based on the molecular graph (nodes being the atoms, and edges being the bonds). The pipeline of our solution based on Graph Neural Networks is the following:

1. preprocessing. File preprocess.py adds more features to the file structures.csv (IsInRingSize() returns whether the atom is in a ring of a particular size or not).
2. triplet_update. For updating so-called triplet features (angles, areas of triangle) 4 fully-connected layers net has been used. It uses concatenated input of several features (atom types, charges, distances, bond types, angles, areas of triangle) and returns the updated area values of triangles as an output for each iteration.
3. node_edge_update. Both graph nodes and graph edges hold feature vectors. Size of the node and edge feature vectors are equal in each layer. For updating nodes and graph edges, it holds feature vectors and 4 fully-connected layers net have been used. It uses concatenated input of several features (atom types, charges, distances, bond types, angles, areas of triangle) and returns updated values of nodes and graph edges.
4. submission. File submission.py concatenates all csv-files from steps 2 and 3 and creates the final submission.

3. Results

3.1. Decision trees

The main advantages of tree-based model are speed and ease of writing and interpretation. We spent just 1 hour using laptop with Core i7 8 gen to train the model. Our final score was: logMAE = -1.3932970243725193. The main drawback of this model is its inability to achieve very good results. The model can be used as a baseline. For more accurate results, it is necessary to use graph neural networks.

3.2. Graph Neural networks

For training, AWS credits were used. We spent around 10 hours setting up a virtual environment for computing. It took 12 hours to debug and test the model. It should be noted that the model is very computationally-intensive. Over a 24-hour period of time, the model has passed 5 epochs of training with the use of one Tesla K80. Our final score was: logMAE = -1.33121. We didn't have enough computing power to run 100 epochs with several GPUs, so this result was quite low due to the small number of epochs. Teams which used GNN during the competition, got quite good results with the use of GNN but with much bigger number of epochs. For example, top-8 teams used different GNN architectures with at least 50 epochs.

The graph of Log Mean Absolute Error for 5 epochs is presented below:

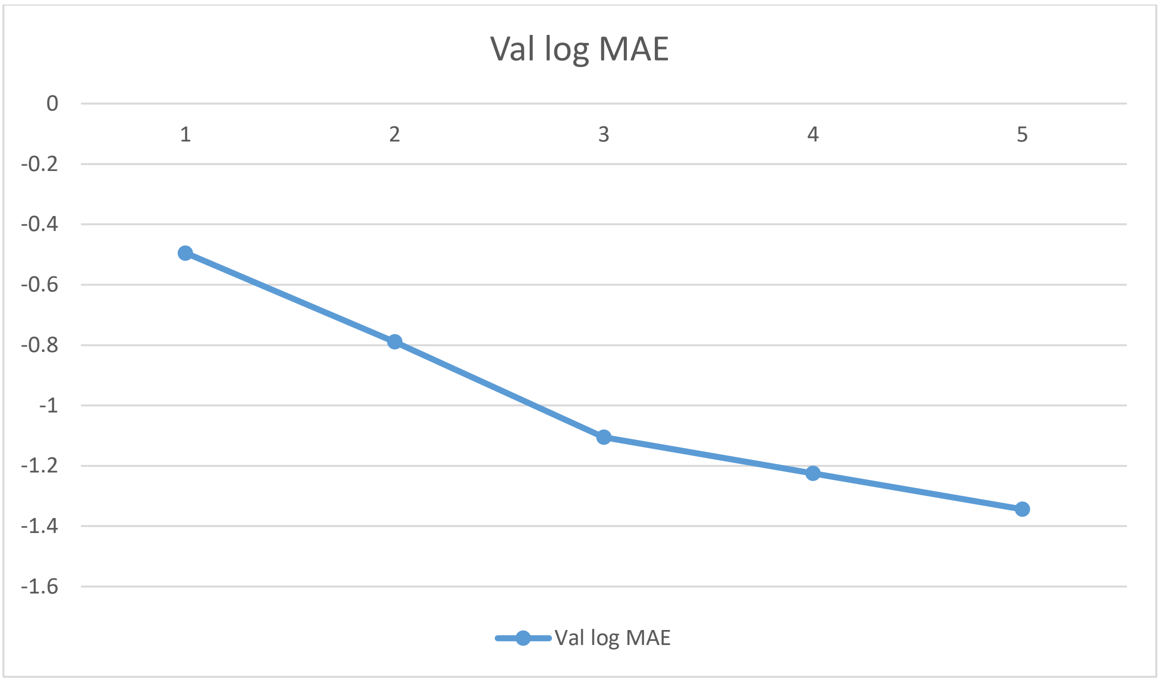

Figure 3-1. Log Mean Absolute Error for 5 epoches for Graph Neural Network.

4. Conclusion

During our research we got an interesting new experience working with graph neural nets.

We used 2 different methods: the first with the using of tree-based model (LightGBM), the second — with graph neural model (Chainer). Tree-based model is very easy to use but has limitation in its predictive power. Graph neural model are much more effective for predicting molecules properties but they are very computationally-extensive. We set up the model but did not get good results due to limited computing resources.

5. References

https://github.com/toshi-k/kaggle-champs-scalar-coupling — this repository was used as reference to create graph neural network for predicting scalar coupling constants

https://github.com/jensengroup/xyz2mol — this repository was used as library for molecular scructure parsing

https://www.kaggle.com/asauve/v7-estimation-of-mulliken-charges-with-open-babel — this link was used to get mulliken charges with with Open Babel library

https://github.com/pfnet-research/chainer-chemistry - Chainer Chemistry: A Library for Deep Learning in Biology and Chemistry

6. Appendix

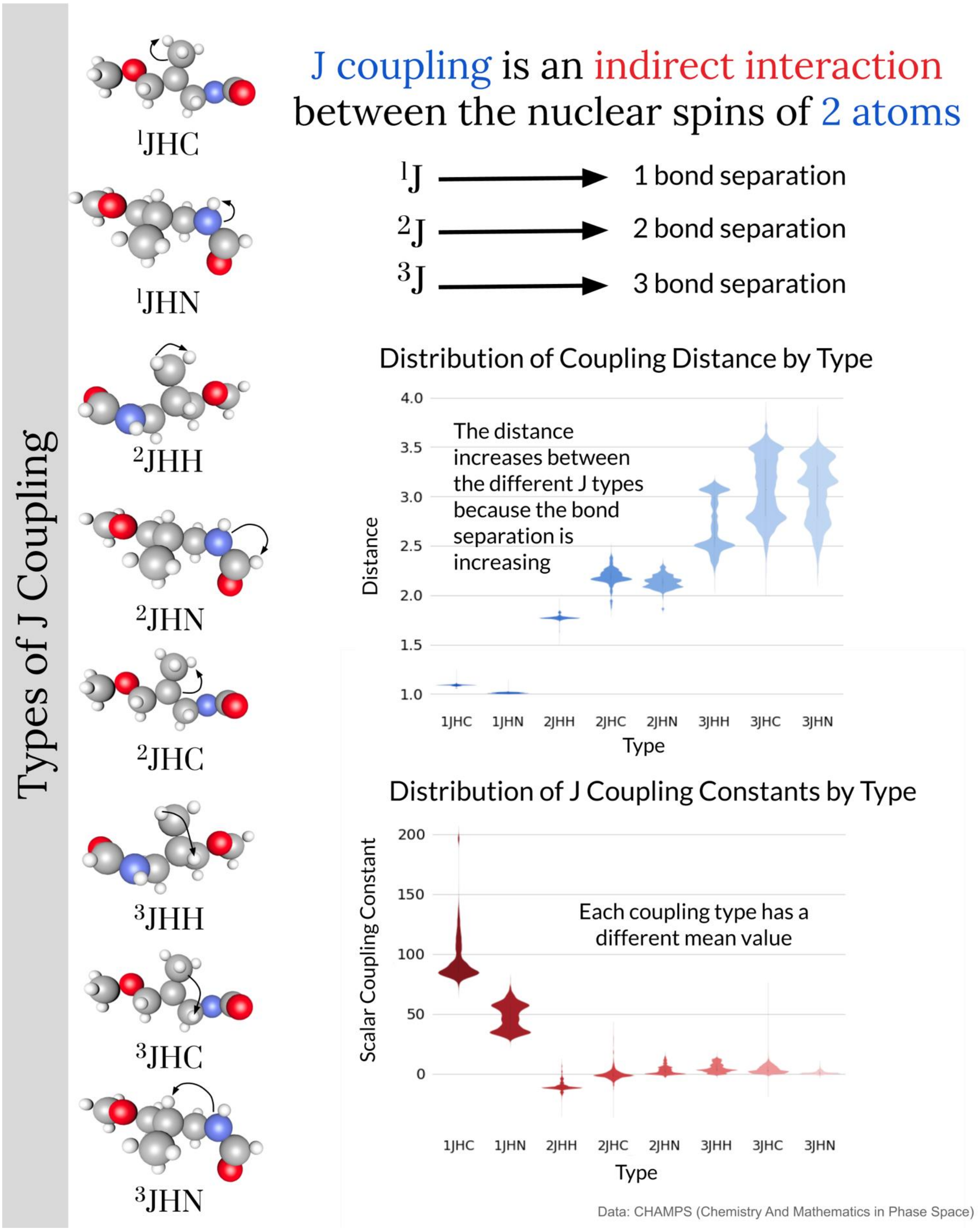

Figure 6-1. Explanation of coupling distances

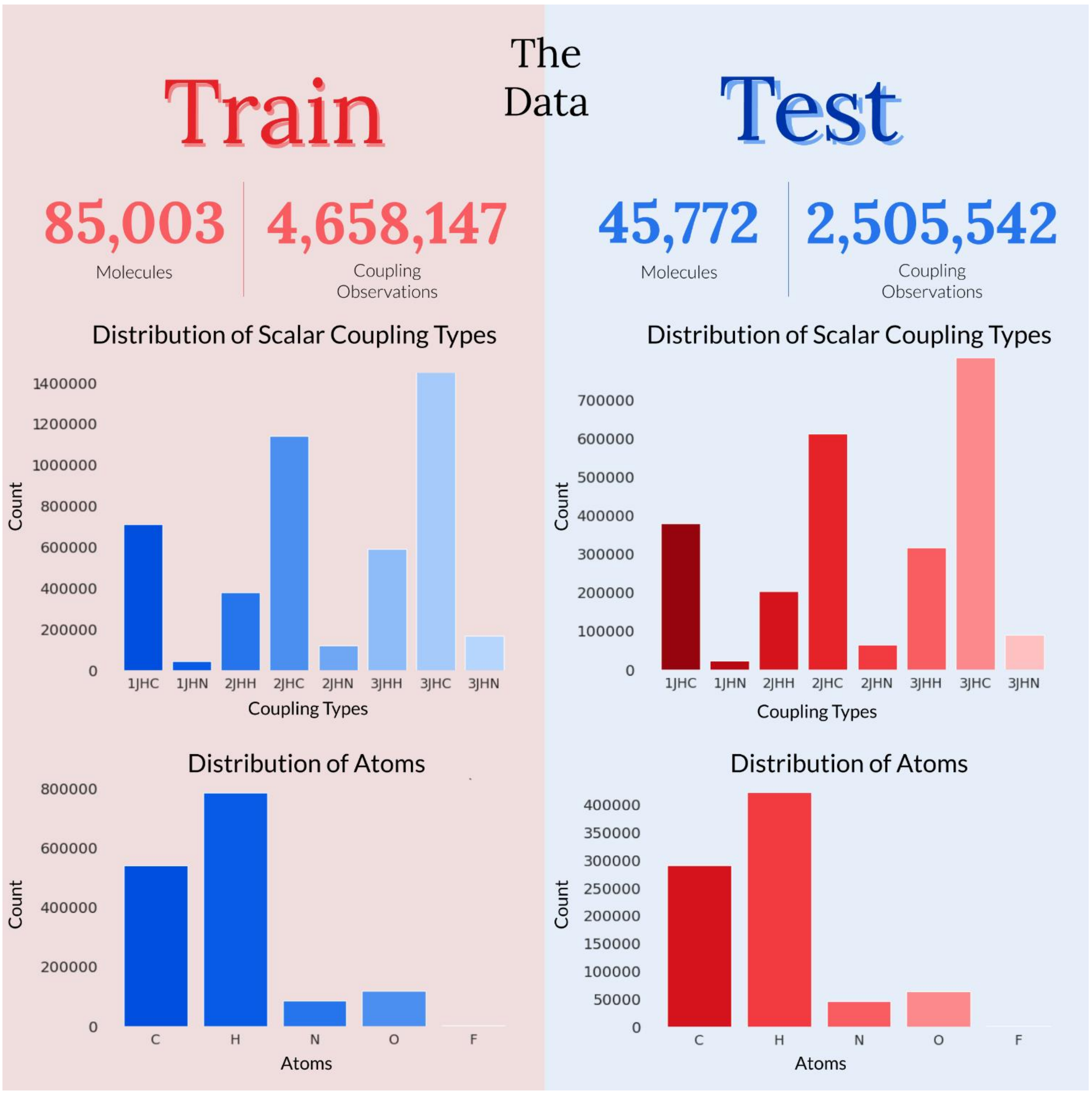

Figure 6-2. Exploratory data analysis

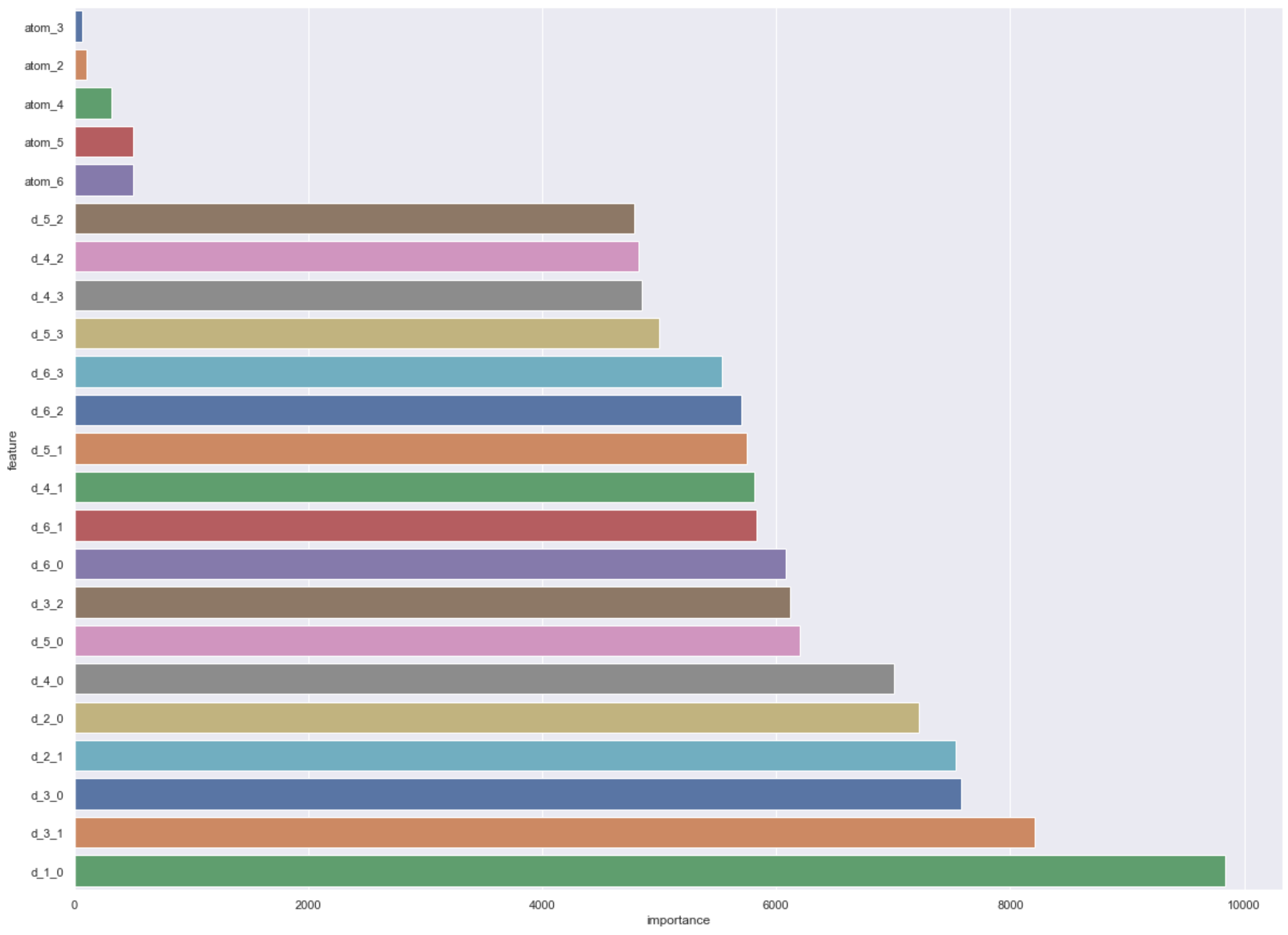

Figure 6-3. Feature importance for tree-based model