Capstone 1 Predicting Molecular Properties

Jingling Zhang

# Introduction

## Background

Nuclear Magnetic Resonance (NMR) spectroscopy is an analytical chemistry technique used in quality control and research for determining the content and [purity](http://chem.ch.huji.ac.il/nmr/qc.htm#impurities) of a sample as well as its [molecular structure](http://chem.ch.huji.ac.il/nmr/identify.htm). It shed light upon determination of molecular conformation in solution as well as study of physical properties at the molecular level such as [conformational exchange](http://chem.ch.huji.ac.il/nmr/techniques/other/dynamic/dynamic.html), phase changes, solubility, and [diffusion](http://chem.ch.huji.ac.il/nmr/techniques/other/diff/diff.html).

Using NMR to gain insight into a molecule’s structure and dynamics depends on the ability to accurately predict ***J*-couplings** or **Scalar couplings**, which contains information about relative bond distances, angles and connectivity of chemical bonds. *J*-couplings reflect 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.

It is possible to accurately calculate *scalar* coupling constants (*J*) 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 and analytical 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.

## Project focus and potential clients

The focus of this Capstone Project would be to develop an algorithm that predict the scalar coupling constant. This project would help analytical chemists, pharmacologists, and physicists to better understand the fundamentals of chemical structures and relevant molecular properties. It could also give insights into streamlining chemical synthesis and characterization in industry.

# Data Wrangling

## Description of dataset

Data was provided by Chemistry and Mathematics in Phase Space (CHAMPS) at the University of Bristol, Cardiff University, Imperial College and the University of Leeds (<https://www.kaggle.com/c/champs-scalar-coupling>). Because the purpose of this capstone project is to generate a model for predicting scalar coupling constant based on 3D structures of given molecules, only *train.csv*, *structures.csv* and the *structures* folder containing xyz files would be used. Other csv files such as *potential\_energy.csv* and *dipole\_moments.csv* are not relevant in this project.

**train.csv**: for training and testing models; the first column (molecule\_name) is the name of the molecule where the coupling constant originates, the second (atom\_index\_0) and third column (atom\_index\_1) are the atom indices of the atom-pair creating the coupling, and the fourth column (scalar\_coupling\_constant) is the scalar coupling constant that we want to be able to predict.

**structures.csv**: used to extract information for training model; consist of molecule\_names, atom indices, atoms, X, Y and Z cartesian coordinates.

**Structures folder**: folder containing molecular structure (xyz) files, where the first line is the number of atoms in the molecule, followed by a blank line, and then a line for every atom, where the first column contains the atomic element (H for hydrogen, C for carbon etc.) and the remaining columns contain the X, Y and Z cartesian coordinates (a standard format for chemists and molecular visualization programs)

## Data cleaning

Two csv files (train.csv and structures.csv) were imported as DataFrames using pandas, namely *train* and *structures* respectively. Two DataFrames did not have null or duplicated values; the column names were consistent, and column data types were reasonable.

However, *structures* contained more rows than did *train*, because in the original Kaggle competition *structures* also included information for molecules in *test.csv* (used for testing machine learning model and project submission). In this project, 90% randomly selected data from train would be used to train models and the remaining 10% data would be used to test models. Additionally, in *structure*, there were a small number of molecules containing Fluorine (F), but no atom pairs contained F in *train*, meaning atom pairs containing F would not be covered in this project to predict the scalar coupling constant.

There are lots of outliers in *train*, which is possible as *J* varies dramatically for different molecules. Outliers in the case should not be removed. Overall data appeared in the right range, for example, all one-bond couplings had positive *J*, and most of 3*JHH* couplings had positive constant, with a small percentae having arbitrarily small negative values.

## Data wrangling

As the scalar coupling is a through-bond interaction, angles between atoms and distances between atoms could be significantly impact *J* (Figure 1). 2-bond coupling constants 2*J* can be very different for geminal protons (2*JHH)*, depending on the hybridization of their mutual carbon. For unstrained sp3 CH2 protons with innocuous substituents, the coupling is typically around ~12 Hz, whereas the 2-bond coupling of sp2 protons is much smaller. The hybridization could be deduced from the bond angle of H-C-H, for example, sp3 usually has the bond angle of ~109.5° and sp2 usually has the bond angle of 120°. Therefore, generating data of bond angle and distance between atoms could possibly provide more information for predicting *J*.

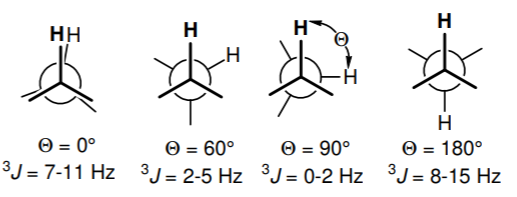
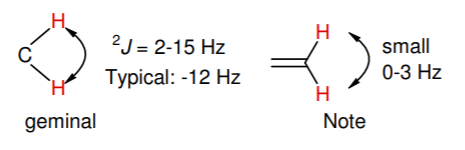


Figure Empirical scalar coupling constants J

The calculation of distance between 2 atoms was based on cartesian coordinates provided in *structures*. After merging *train* and *structures* (inner join) into a new DataFrame *final*, calculation of distance between 2 points with cartesian coordinate values was done referencing https://www.kaggle.com/seriousran/just-speed-up-calculate-distance-from-benchmark, and results were stored in a new column “distance” of *final*.

The calculation of bond angles was very tricky and time consuming. RDKit and xyz2mol were used to import xyz files in *structures* folder as mol files (<https://github.com/jensengroup/xyz2mol/blob/master/xyz2mol.py>) . However, due to the limitation of RDKit dealing with explicit valence for nitrogen(N), there were 366 xyz files/ molecules could not be imported, and 234 of them were present in *final*, with 11058 relevant entries. There were 85012 molecules and 4659076 entries in total in *final*, of which the inaccessible xyz files take a very small part. The explicit valence issue for N is a very frequent and unsolved issue for RDKit. Therefore, the readable molecules were converted to mol files and stored in *mol\_dict* for further calculation of bond angles, and inaccessible xyz files were stored in *fails\_path* temporarily. The unreadable molecules were temporarily removed from *final* until a solution is found, and the resulting DataFrame was named *filtered*.

Using GetAngleDeg() from rdkit.Chem.rdMolTransforms allowed us to calculate angles between 3 atoms. In *filtered*, 2 terminal atoms were known as atom\_index\_0 and atom\_index\_1. To calculate the bond angle, one would have to know the common neighbor of these 2 terminal atoms. For example, for the chemical structure below (Figure 2, left) H1 and H2 would correspond to atom\_index\_0 and atom\_index\_1, C0 must be located before calculating the angle (marked with arrow).



Figure 2 Examples for calculating bond angles

It is relatively easy to locate the neighboring atom in the case of ***2J*** coupling. Since H atom only has 1 covalent bond, the only neighboring atom that H atom has is the common neighbor between two terminal atoms (atom\_index\_0 and atom\_index\_1). Functions getdegree() and degree\_2J() based on GetAngleDeg() were used to find the neighbors of the one and only neighbor of H atom in an atom pair. Bond angles were calculated as the angle mean for all conformers of the targeted molecules, and the resulting values were stored in ”degree” column in *filtered.*

For ***3J*** coupling, GetDihedralDeg() would be used instead of GetAngleDeg(). GetDihedralDeg() is the method to calculate dihedral angle in degrees between 4 atoms. 2 terminal atoms were already given as atom\_index\_0 and atom\_index\_1. The case for *3JHH* is easy as H atoms could only have one neighboring atom at most. For example, to find the bond angle of the above chemical structure (Figure 2, right), we have to identify the neighboring atoms of H2 and H3 first and then the bond angle could be calculated based on coordinates of these four known atoms. This is the basic idea behind the function degree\_3JHH(). Scenarios for *3JHC* and *3JHN* are a little more difficult: first we need to find which atom in the given atom pairs is hydrogen (for example, atom\_index\_0 is H), and we need to find the immediate neighboring atom (*immediate0*) of this hydrogen; then we list out all neighbors of *immediate0* in a list called *secondary*; finally we loop over all elements in *secondary* and find each element’s neighboring atom. If the neighboring atom is atom\_index\_1, then we find the four atom that are linked by three chemical bonds. Degree\_3J() and find\_neighbors() are used for this part of calculation. The absolute values are extracted for negative bond angles.

For 1*JHH* and 1*JHN*, the angle was broadcast as 0 in “degree” column; dihedral angles for *3JHC*, *3JHH* and *3JHN* were also stored in “degree” column.

There are also a couple instances for *3J* coupling that the bond angle ends up with NaN. This is because that coupling bonds are on the same plane and do not have dihedral angles. Therefore, these NaN values are filled as zeros.

# Exploratory Data Analysis

* 1. Initial finding

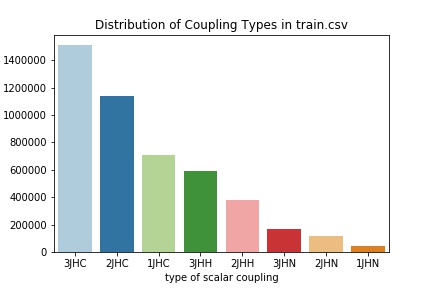


Figure 3 Distribution of coupling types in train.csv

The barplot above shows that three-bond couplings (*3*J) between carbon (C) and hydrogen(H) were the most common coupling type, and one-bond coupling (1J) between hydrogen(H) and nitrogen(N) was the rarest in train.csv.

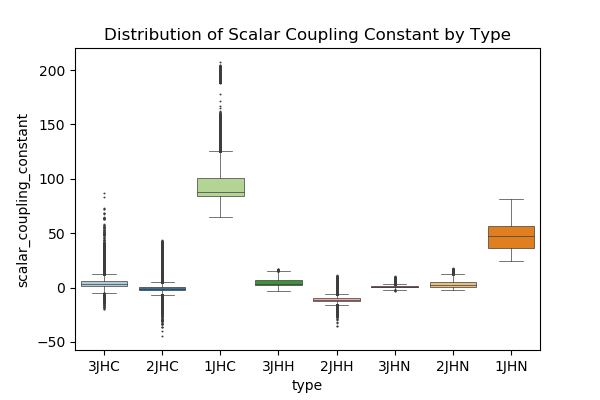


Figure 4 Distribution of coupling types in train.csv

All one-bond couplings including 1*JHC* and 1*JHN* had positive scalar coupling constant as expected, and for most of 3*JHC*, scalar coupling constants were positive or very close to 0. It appears that one-bond coupling 1*JHC* had the highest overall magnitude, followed by 1*JHN.*

In structures.csv, most of molecules consisted of 15-21 atoms, small molecules containing less than 10 atoms were not common in this dataset (Figure 5).

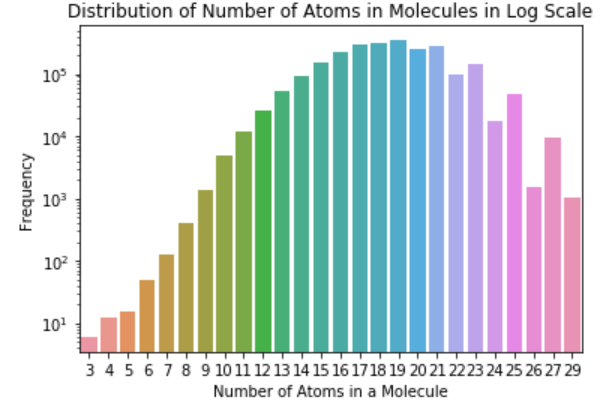


Figure 5 Distribution of number of atoms in molecules from structures.csv

Figure 6 shows that in DataFrame *final* the atom pairs of three-bond coupling were usually more far apart than those of one-bond coupling. Generally, C-H bonds are shorter than N-H bonds. But in final, most of 1*JHN* had smaller distance than 1*JHC*. Figure 6 together Figure 4 showed that longer distances between atoms would possibly be related to smaller *J*.

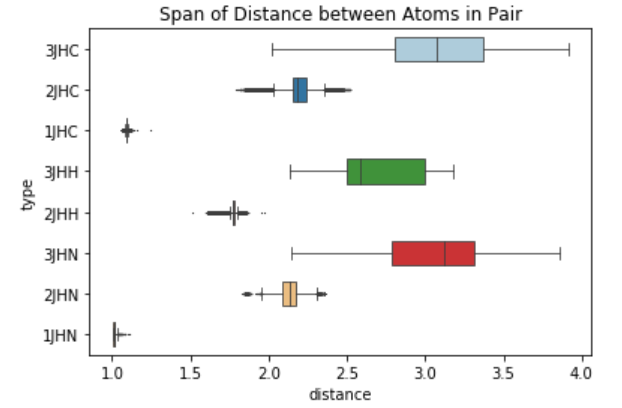


Figure 6 Distribution of distance between atoms in pair from DataFrame final

Using rdkit library can help visualize chemical structures of molecules. 9 molecules were randomly selected, and their chemical structures were shown as following (Figure 7).

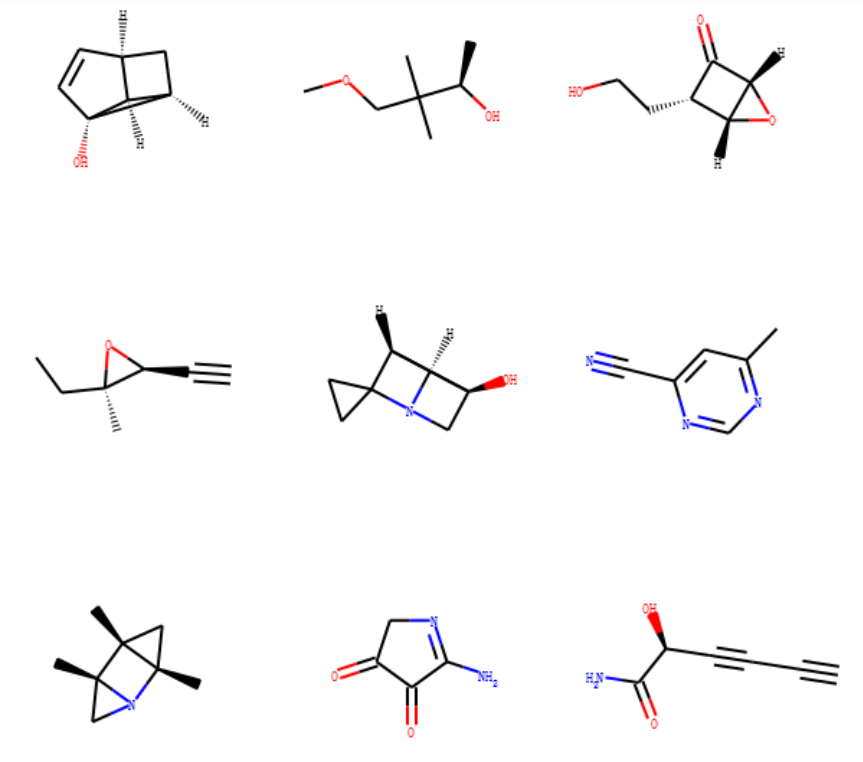
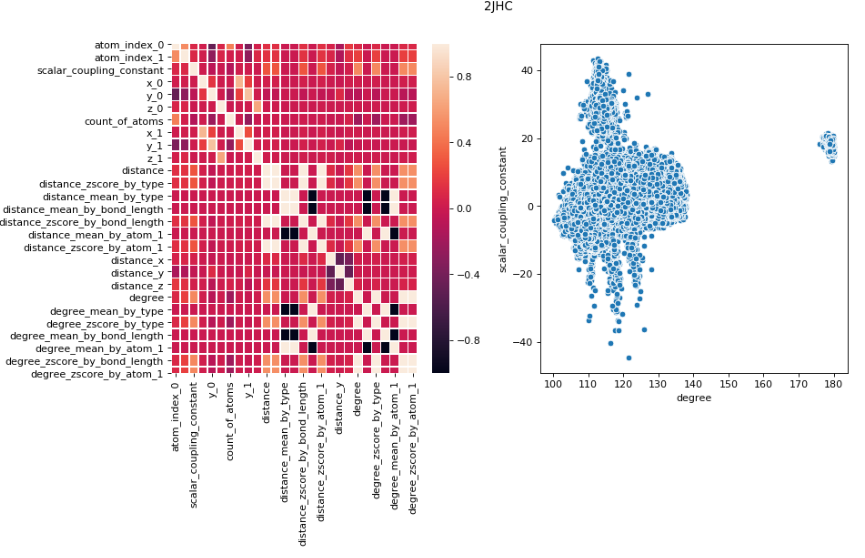
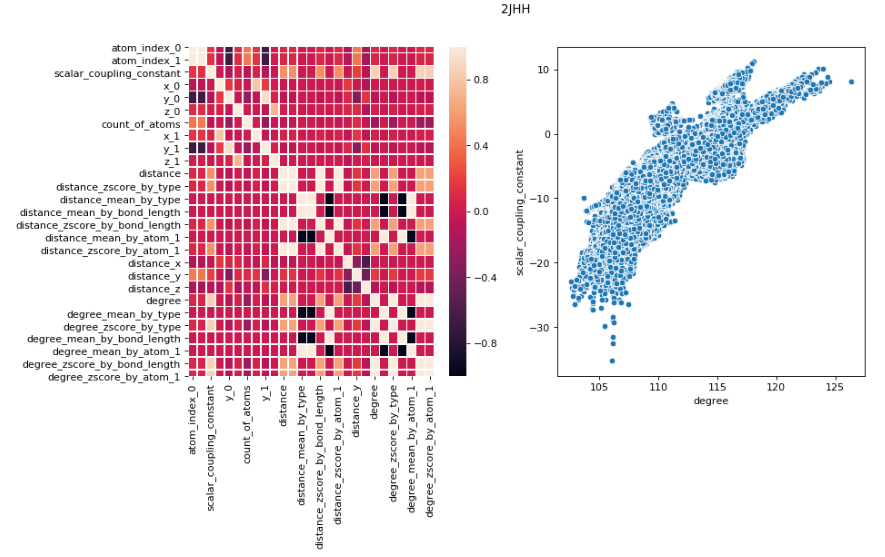


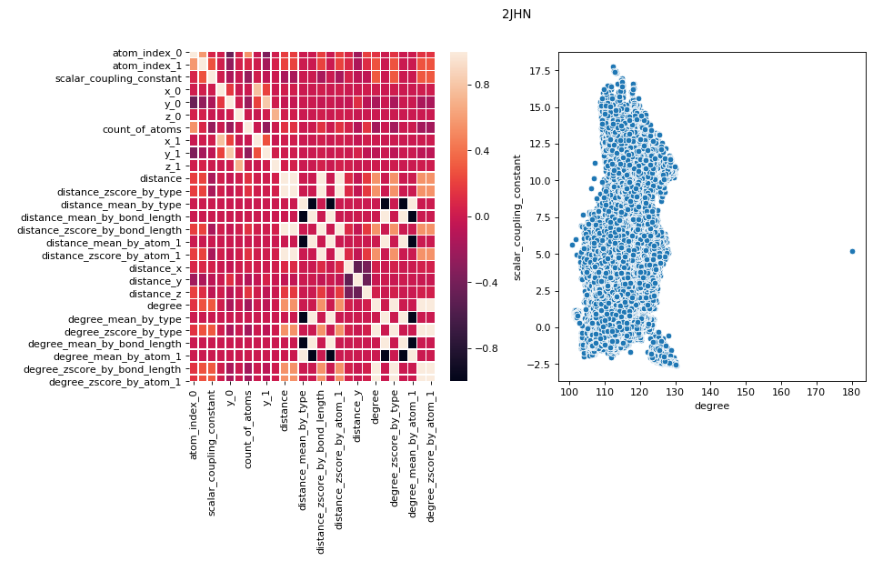
Figure 7 Chemical structures of 9 randomly selected molecules from structures folder

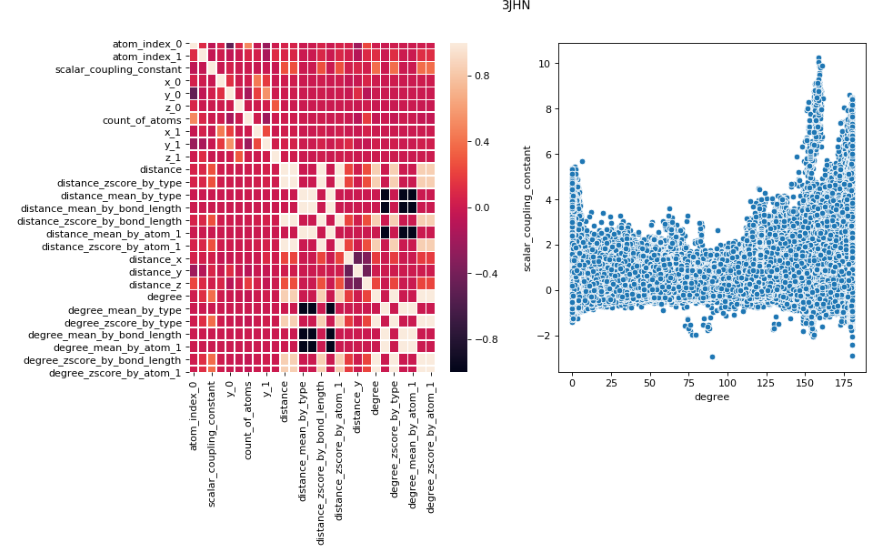
Heatmaps and scatterplots are generated for each type of coupling as following. The relationship between bond angle and scalar coupling constant is almost linear in the case of 2*JHH*. For *3JHN* and *3JHH*, the distribution has a nice U shape. The distribution for other types is not as unclear but still has some interesting pattern.

From the heatmaps, we can see that scalar coupling constants for some types are strongly related to distance and bond angle.









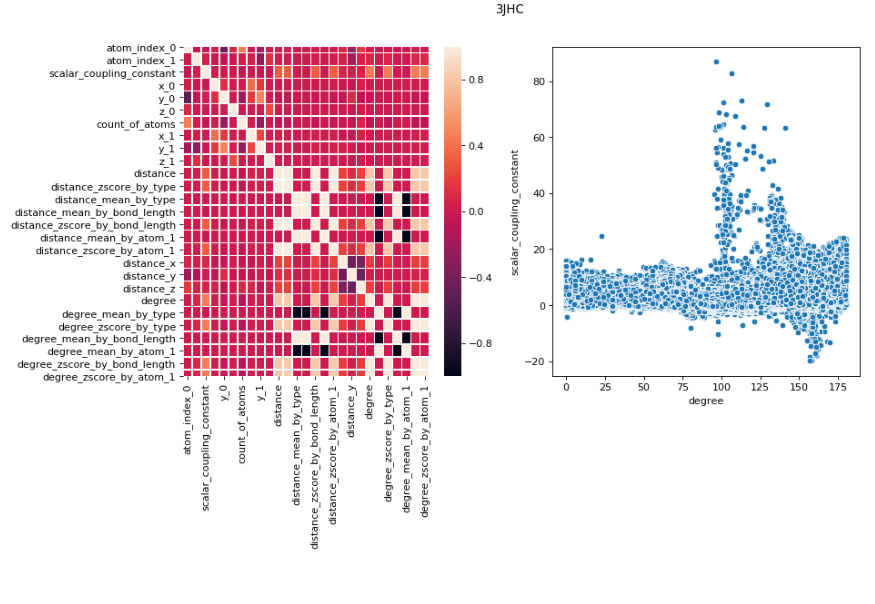
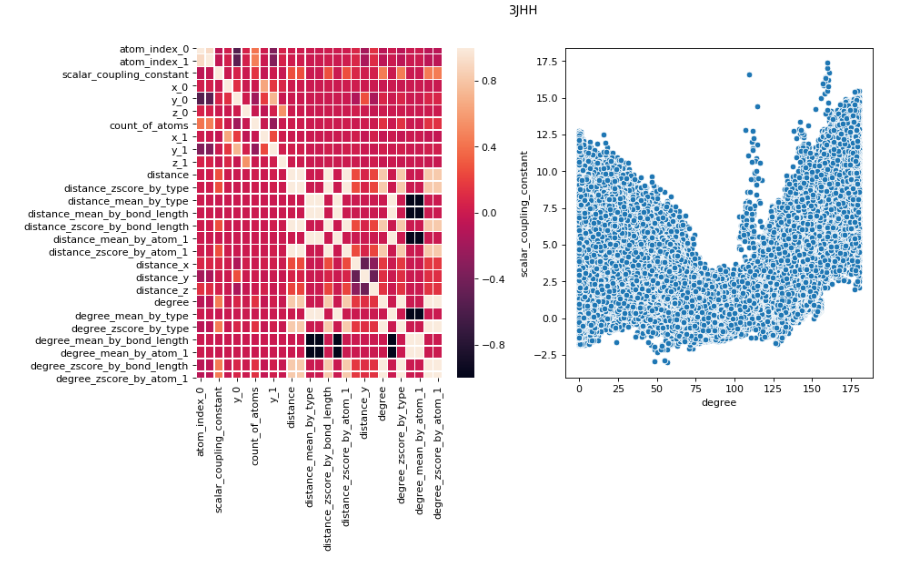
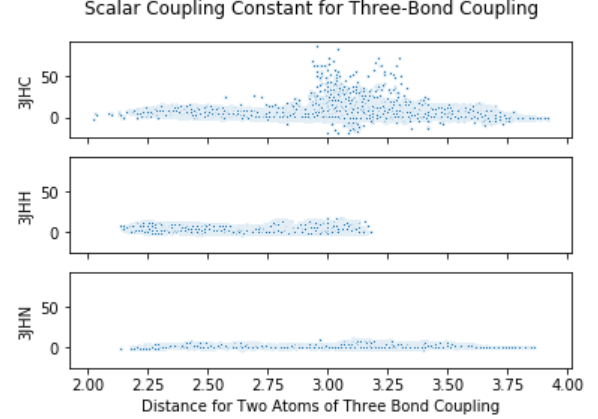
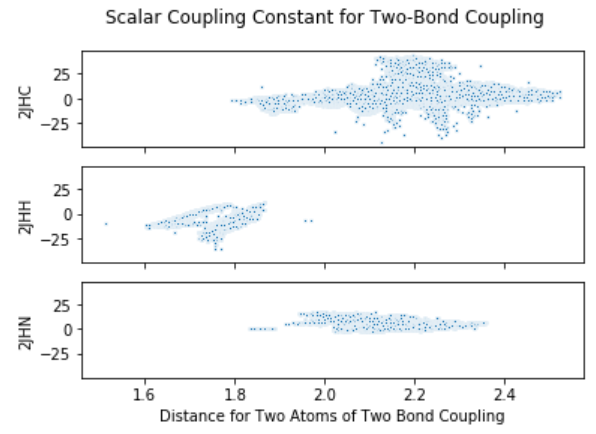
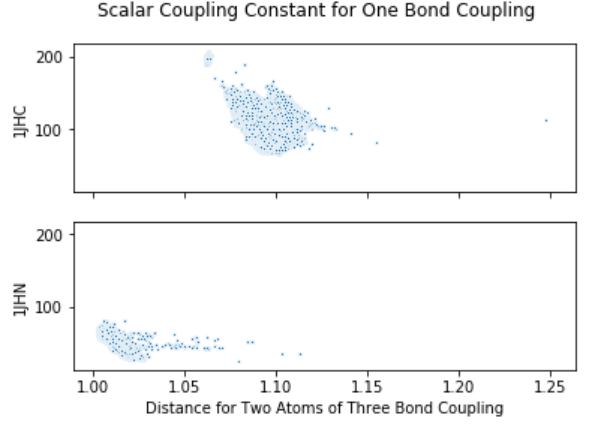


Figure 8 Heatmap of Pearson R correlation and scatterplots for all coupling types

## Statistical finding

Scatterplots of different coupling types were shown as below. For 1*J*s, distance and *J* seemed to be negatively correlated. The trends for 2*J*s and 3*J*s were not very consistent throughout types: 2*JHH* seemed to have positive correlation between distance and *J;* it seems like that there was no correlation between distance and *J* for 3*JHH,* 3*JHN and* 2*JHN*.



*Figure 8 Heatmap of Pearson R correlation from DataFrame final*

Permutation tests were done to prove if there was correlation between distance and *J* for 3*JHH,* 3*JHN and* 2*JHN*, respectively. *J* was permutated and distance was kept the same; correlation was studied on permutated *J* and distance (1000 permutated replicates were generated due to save time). H0: There is no correlation between distance and *J*. At alpha = 0.05, H0 was rejected for 3*JHH,* 3*JHN* but was retained for 2*JHN*, meaning there was no statistically significant correlation between distance and *J* for 2*JHN.*

# Data modeling

## Data preprocessing

Final dataframe used for model is called *prepared*. Column `scalar\_coupling\_constant` is used as the target (y) and other selected columns are used as features (X). There are categorical features, so LabelEncoder() is used for transforming categorical data into numerical data. Corresponding labels are stored in a dictionary called *mapping*.

The metric for evaluation (ref <https://www.kaggle.com/uberkinder/efficient-metric>) is a mean absolute error based metric. When all predicted values are 0, the metric has the value of 1.9975. And the smallest possible value for metric is -20.7323.

Data are separated into Xtrain, Xtest, ytrain, ytest using train\_test\_split(). The test data set will only be used once to examine if a model is generalizable to new data.

## Simple Linear Regression Model

Simple linear regression model is used as a benchmark model without regularization. This model does not overfit much but has low capacity for accurate prediction.

## KNN Regression Model

Principal components analysis is done first to reduce dimensionality. It is shown that more than 70% variance could be explained by 5 features. The dimensionality is reduced to 6.

K nearest neighbor regression is tuned and *n\_neighbors = 5* gives the least error. This model has better capacity for prediction but overfit.

## Decision Tree Regression Model

Decision Tree Regression is used without utilizing PCA. It is much faster than KNN model and only take about 2min to run. After tuning hyperparamters including *max\_features* and *min\_sample\_split*, this model has better predicting power and less overfitting problems.

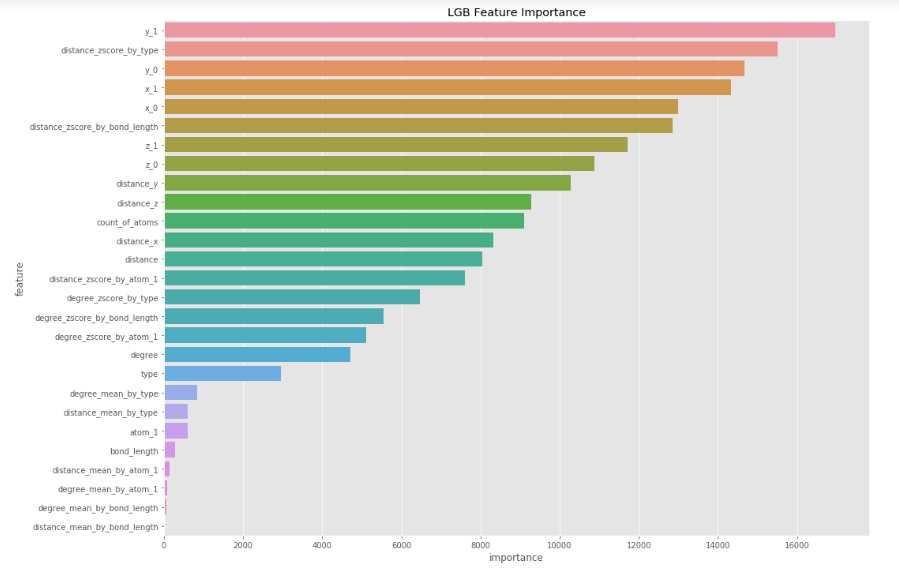
## Light Gradient Boost Machine Model

Light GBM is used and hyperparamters are tuned in the following steps:

1. fix learning rate and number of estimators for tuning tree-based parameters
2. tune num\_leaves and min\_data\_in\_leaf
3. tune min\_gain\_to\_split
4. tune bagging\_fraction + bagging\_freq and feature\_fraction
5. tune lambda\_l2

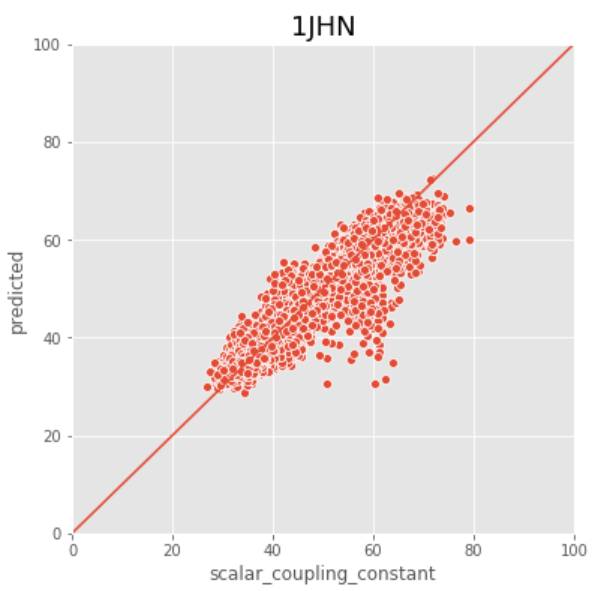
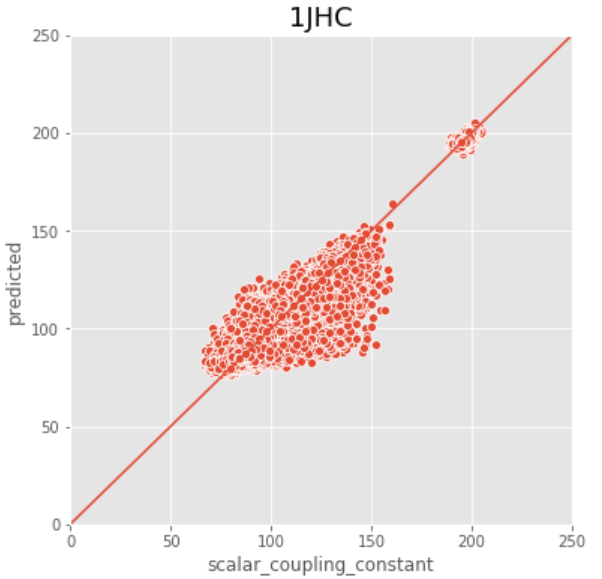
This model needs a lot of computation power (takes approximately 1 hour to run 10000 trees), but after tuning it gives more accurate prediction and way less overfitting issue. The accuracy of prediction could be improved by having large *n\_estimators* but will also be more time consuming. Limited to my present computation power, only *n\_estimators = 10000* is used here.

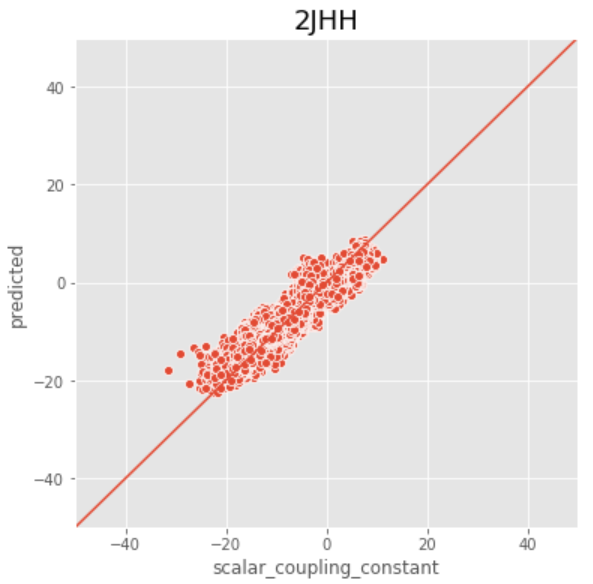
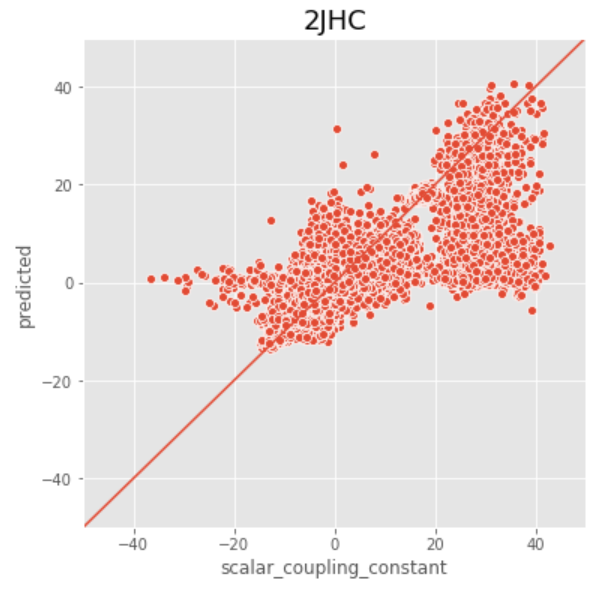
Most important features are plotted as below.

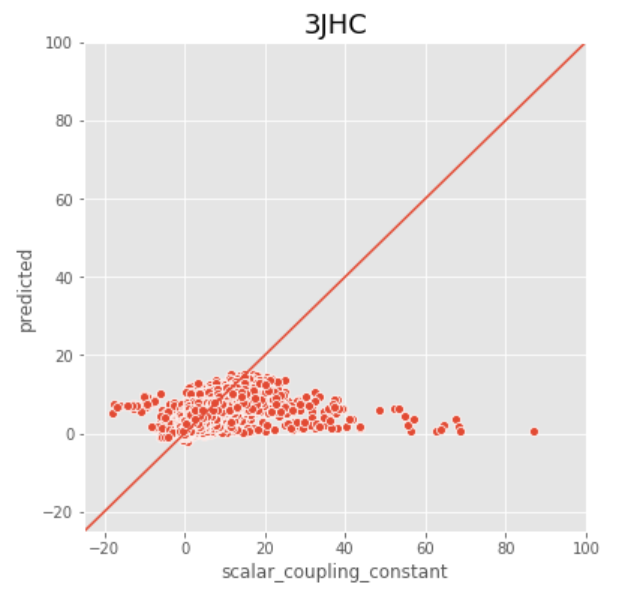
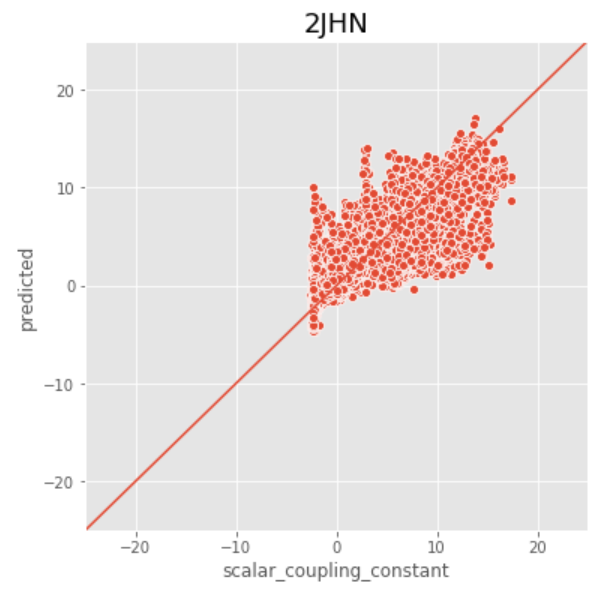


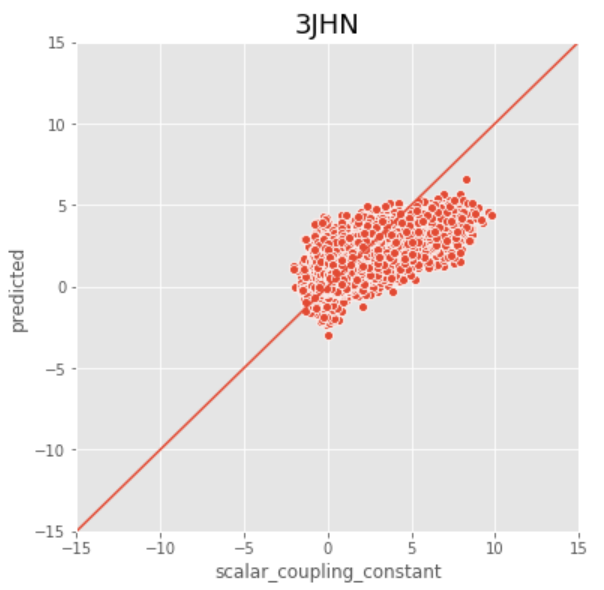
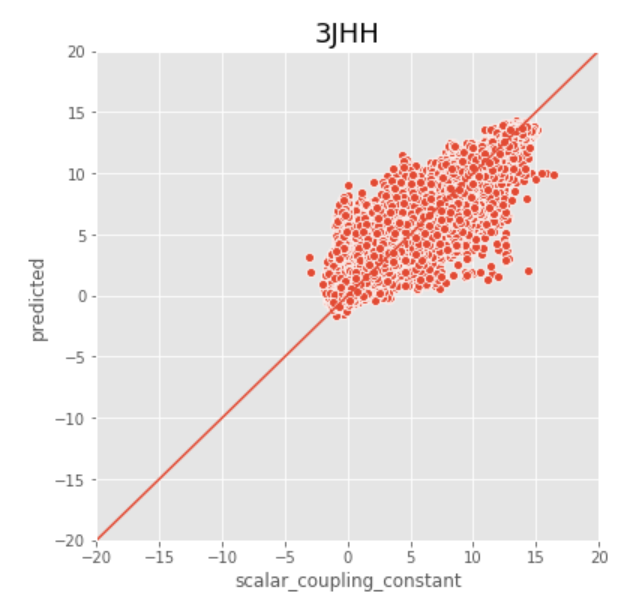
*Figure 9 Importance of features in Light GBM model*

Original values vs predicted values are plotted for each type of coupling as below. For most of the types, the predicted values agree well with original values. However, for *3JHC* and *2JHC* the model seems to suffer from low accuracy.









*Figure 10 Original values vs predicted values in Light GBM model*