Lab ML For Data Science: Part II

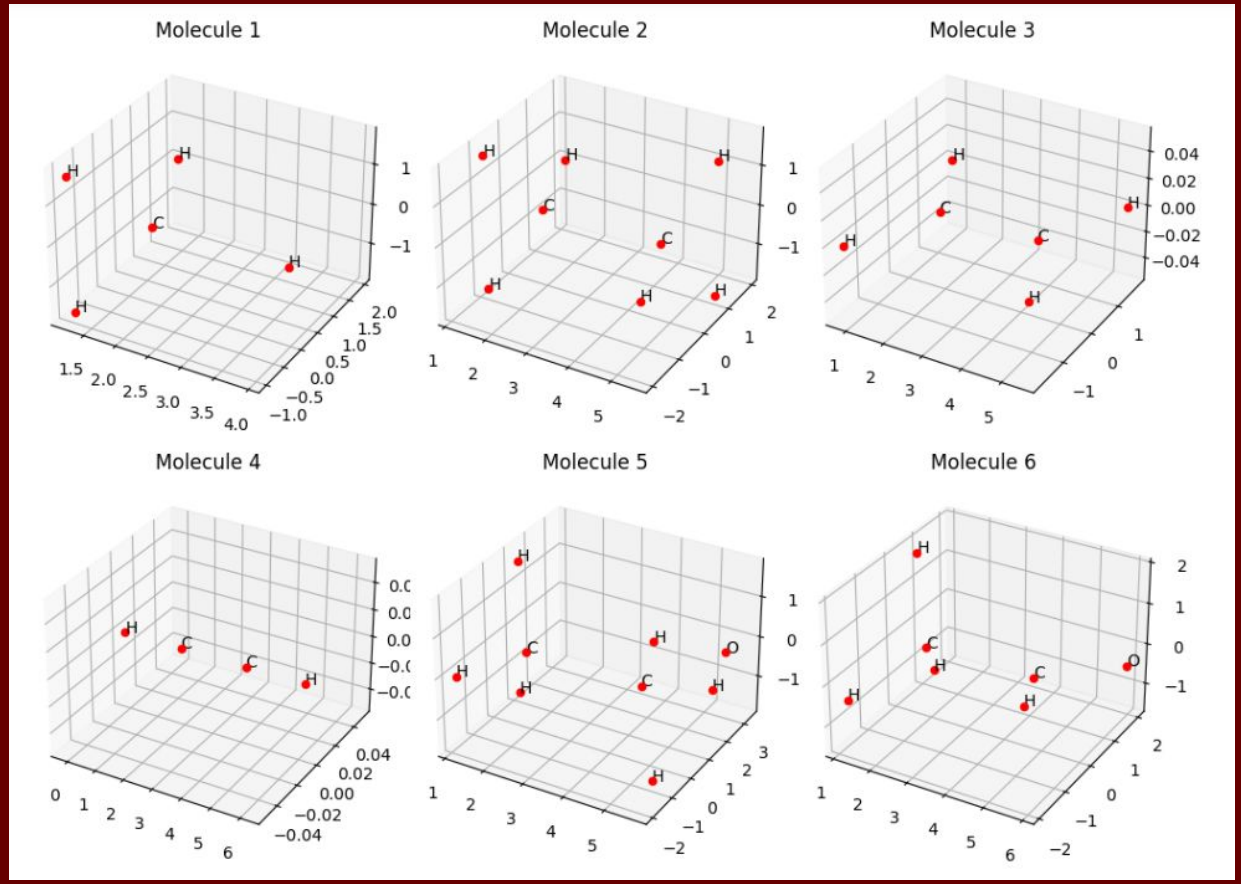
Getting Insights into Quantum-Chemical Relations

1.1.

Visualizing Molecules

QM7 DataSet

- Consists of 7165 molecules and its atomic structure, each molecule comprises 23 atoms
- Variable R:
  - It has the 3D coordinates of each of the 23 atoms in a single molecule.
  - Shape: 7165x23x3
  -
- Variable Z:
  - It has the atomic number of each atom in a molecule.
  - Shape: 7165x23
  - Atomic Number 0 signifies no atom
- Variable T :
  - It has the atomization energy.
  - Shape: 1x7165
  - Signifies energy required to break bonds



Plain scatter plot of the atoms of first 6 molecules in 3D space

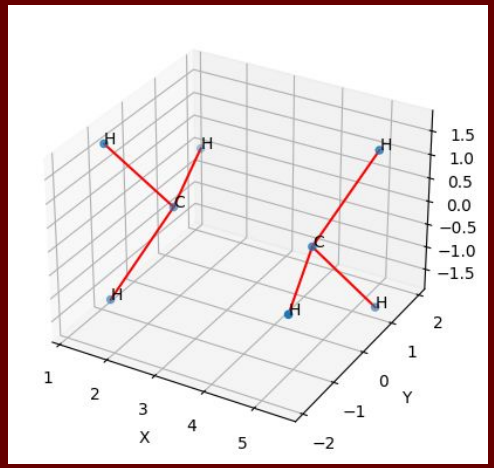
Finding an existing bond

Out of the 23 atom coordinates, first molecule has 5 atoms.
To find out an existing bond, based on a distance threshold, we consider only the non-zero coordinates.
We form a list of 7165 molecules having the shape: (Number of atoms)x3

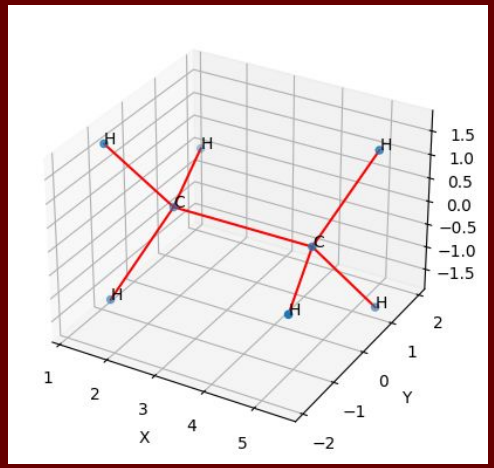
```
First molecule has only 5 valid atoms, and the coordinates are given by :

array([[ 1.886438  , -0.00464873, -0.00823921],
       [ 3.9499245 , -0.00459203,  0.00782347],
       [ 1.1976895 ,  1.9404842 ,  0.00782347],
       [ 1.1849339 , -0.99726516,  1.6593875 ],
       [ 1.2119948 , -0.9589793 , -1.710958  ]], dtype=float32)
```

Plotting the second molecule C2H6

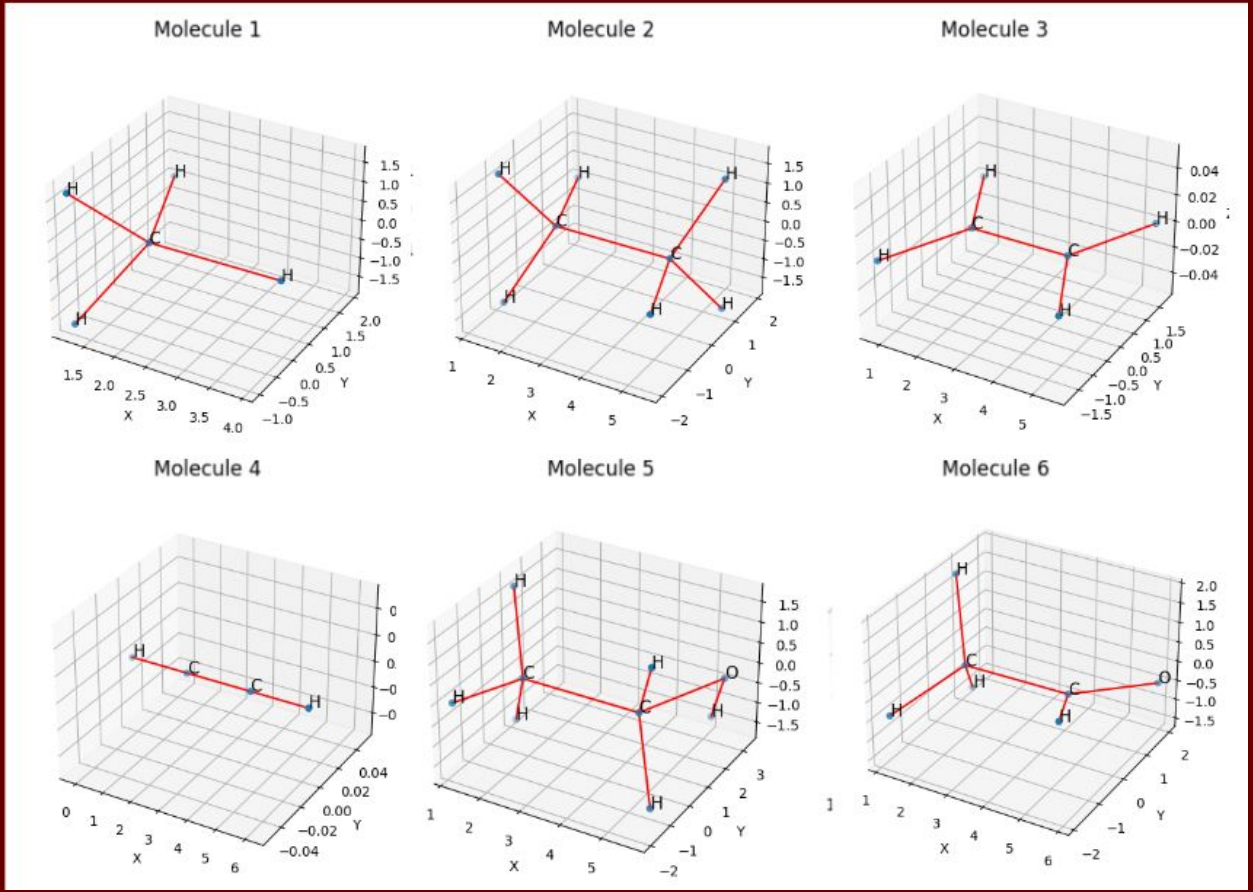
Threshold distance=2.5



Threshold distance=3



We know that in C2H6, there is a bond between the two Carbon atoms. So we decided to move the distance threshold to 3 for a more accurate representation.



Scatterplot of the atoms of first 6 molecules with bond threshold=3

2.1

Data Representation

One-hot Encoding

```
unique_atoms = np.unique(qm7_data_atomic_number, axis=None)
unique_atoms
```

```
array([ 0.,  1.,  6.,  7.,  8., 16.], dtype=float32)
```

- The atomic number array (Z) has 6 unique numbers out of which 0 is invalid representing no atom. (We remove it from the list.)
- To do regression on the atoms, as a function of each unique atom in a molecule, we need to have uniform representation of each molecule.
- Hence we one-hot encode each atom and then sum it over unique atoms.

One-hot Encoding

Example:
In the first molecule (CH4) we have 1 C and 4 H and rest are invalid atoms. Since 0 is invalid representing no atom, we remove it.

```
unique_atoms_list [1.0, 6.0, 7.0, 8.0, 16.0]
```

- Now, the first position of the encoder is for atomic number 1 (H) and the second for 6 (C) and so on
- Therefore CH4 can be represented as:
- Shape: 23x5
- Now we sum over one hot encoding of each atom to get uniform representation by each unique atom

Molecular Structure of first 5 molecules:

```
array([[4, 1, 0, 0, 0],
       [6, 2, 0, 0, 0],
       [4, 2, 0, 0, 0],
       [2, 2, 0, 0, 0],
       [6, 2, 0, 1, 0]])
```

```
[[0, 1, 0, 0, 0],
 [1, 0, 0, 0, 0],
 [1, 0, 0, 0, 0],
 [1, 0, 0, 0, 0],
 [1, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0]]
```

2.2.

Ridge Regression Model

Data Preparation for Ridge Regression Model

```
X=one_hot_df_array # Features: One hot representing molecular structure
Y=qm7_data_atomization_energy.reshape(-1, 1) # Target: Atomization Energy

#Splitting the data in train and test and validation
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42,)
X_train, X_val, y_train, y_val   = train_test_split(X_train, y_train, test_size=0.25, random_state=1)

# Scaling after splitting
X_train_scaled= scaler_X.fit_transform(X_train)
X_test_scaled= scaler_X.transform(X_test)
X_val_scaled = scaler_X.transform(X_val)

y_train_scaled=scaler_Y.fit_transform(y_train)
y_test_scaled=scaler_Y.transform(y_test)
y_val_scaled = scaler_Y.transform(y_val)
```

Determining the Regularization Parameter (lambda)

We optimize the regularization parameter based on the improvement in mean squared error.
We get the optimised lambda to be 0.28
After running the Ridge model, we get the weights as:

```
array([[-0.78542092, -0.63865221, -0.36023638, -0.33025999, -0.07239034]])
```

```
MSE: 0.008387
Lambda: 0.01
MSE: 0.0083869
Lambda: 0.05
MSE: 0.0083868
Lambda: 0.13
MSE: 0.0083867
Lambda: 0.2
MSE: 0.0083866
Lambda: 0.28
Best MSE: 0.0083866
Best Lambda: 0.28
```

Predictions from ridge seem to match the actual values (both scaled) with the following error values:

```
Maximum error between predicted and actual atomization energy: 0.2697
Average error in predicted and actual atomization energy: -0.0049
```

Closed Form Solution

- The Ridge weights can be calculated using the closed form solution given by: $w = (\Sigma_{xx} + \lambda I)^{-1}\Sigma_{xt}$

where $\Sigma_{xx} = \mathbb{E}[xx^\top]$ and $\Sigma_{xt} = \mathbb{E}[xt]$ are auto- and cross-covariance matrices respectively.

Weights from closed form solution:

```
array([[-0.7853891 ],
       [-0.63850458],
       [-0.36009454],
       [-0.33012226],
       [-0.07235176]])
```

Weights from Ridge Regression Model:

```
array([[-0.78542092, -0.63865221, -0.36023638, -0.33025999, -0.07239034]])
```

2.3.

Deeper Insights with Explanations

Contribution of each atom in Prediction:

```
oh= onehot_encoding[molecule_index]
oh
```

```
[[0, 0, 1, 0, 0],
 [0, 1, 0, 0, 0],
 [0, 1, 0, 0, 0],
 [0, 1, 0, 0, 0],
 [0, 1, 0, 0, 0],
 [0, 0, 0, 1, 0],
 [0, 0, 0, 1, 0],
 [1, 0, 0, 0, 0],
 [1, 0, 0, 0, 0],
 [1, 0, 0, 0, 0],
 [1, 0, 0, 0, 0],
 [1, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0],
```

```
one_hot_df.iloc[molecule_index]
```

```
1.0     5
6.0     4
7.0     1
8.0     2
16.0    0
```

This molecule has 5 H, 4C, 1N and 2 O, i.e. a total of 12 atoms

Multiplying the One Hot representation with our Ridge Regression Weights, gives contribution of each atom:

```
molecule_relevance_by_atom = ridge_weights @ np.array(oh).T
#Relevance of each atom in predicting y (target feature) = w.T . onehotencoding
molecule_relevance_by_atom
```

```
array([[-0.36023638, -0.63865221, -0.63865221, -0.63865221, -0.63865221,
        -0.33025999, -0.33025999, -0.78542092, -0.78542092, -0.78542092,
        -0.78542092, -0.78542092,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ]])
```

The first atom in this molecule is N which contributes -0.360236 to the total atomization energy of -7.5.

Contribution of each atom in Prediction:

	1.0	6.0	7.0	8.0	16.0	relevance
0	0	0	1	0	0	-0.360236
1	0	1	0	0	0	-0.638652
2	0	1	0	0	0	-0.638652
3	0	1	0	0	0	-0.638652
4	0	1	0	0	0	-0.638652
5	0	0	0	1	0	-0.330260
6	0	0	0	1	0	-0.330260
7	1	0	0	0	0	-0.785421
8	1	0	0	0	0	-0.785421
9	1	0	0	0	0	-0.785421
10	1	0	0	0	0	-0.785421
11	1	0	0	0	0	-0.785421
12	0	0	0	0	0	0.000000
13	0	0	0	0	0	0.000000
14	0	0	0	0	0	0.000000
15	0	0	0	0	0	0.000000
16	0	0	0	0	0	0.000000
17	0	0	0	0	0	0.000000
18	0	0	0	0	0	0.000000
19	0	0	0	0	0	0.000000
20	0	0	0	0	0	0.000000
21	0	0	0	0	0	0.000000
22	0	0	0	0	0	0.000000

$$
\begin{aligned}
f(x) &= w^T \cdot (Xscaled) \\
&= w^T \cdot (X - X_{\text{mean}}) \\
&= (w^T \cdot X) - (w^T \cdot X_{\text{mean}}) \\
&= (w^T \cdot X) - AverageAtomizationEnergy \\
Also, \\
f(x) &= w^T \cdot \left(\sum (\text{one hot encodings}) - X_{\text{mean}}\right) \\
&= w^T \cdot \sum (\text{one hot encodings}) - w^T \cdot X_{\text{mean}} \\
&= \sum \left(w^T \cdot (\text{one hot encodings})\right) - w^T \cdot X_{\text{mean}} \\
&= \sum \left(w^T \cdot (\text{one hot encodings})\right) - AverageAtomizationEnergy
\end{aligned}
$$

We can show that $w^T \cdot X$=∑($w^T \cdot$ one hot encodings).

```
ridge_weights @ X[5732]  # y for 5732nd molecule = w.T . X
```

```
array([-7.50246979])
```

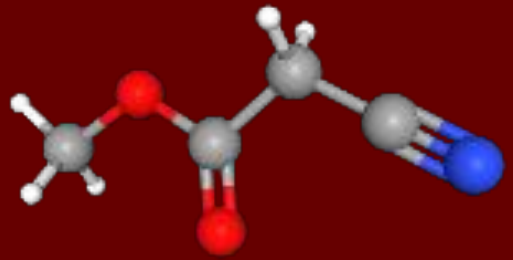
```
(ridge_weights @ np.array(oh).T).sum() #Sum of each individual atom's contribution = Sum(w.T . onehotencoding)
# which is same as w.T . X which we see in the cell above
```

```
-7.502469785542177
```

Contribution of each atom in Prediction:

	1.0	6.0	7.0	8.0	16.0	relevance
0	0	0	1	0	0	-0.360236
1	0	1	0	0	0	-0.638652
2	0	1	0	0	0	-0.638652
3	0	1	0	0	0	-0.638652
4	0	1	0	0	0	-0.638652
5	0	0	0	1	0	-0.330260
6	0	0	0	1	0	-0.330260
7	1	0	0	0	0	-0.785421
8	1	0	0	0	0	-0.785421
9	1	0	0	0	0	-0.785421
10	1	0	0	0	0	-0.785421
11	1	0	0	0	0	-0.785421
12	0	0	0	0	0	0.000000
13	0	0	0	0	0	0.000000
14	0	0	0	0	0	0.000000
15	0	0	0	0	0	0.000000
16	0	0	0	0	0	0.000000
17	0	0	0	0	0	0.000000
18	0	0	0	0	0	0.000000
19	0	0	0	0	0	0.000000
20	0	0	0	0	0	0.000000
21	0	0	0	0	0	0.000000
22	0	0	0	0	0	0.000000

One possible combination with this atomic structure is $C_4H_5NO_2$

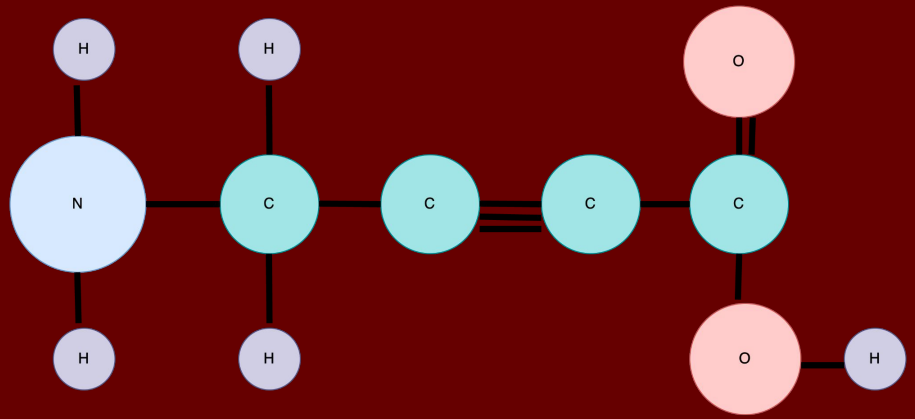


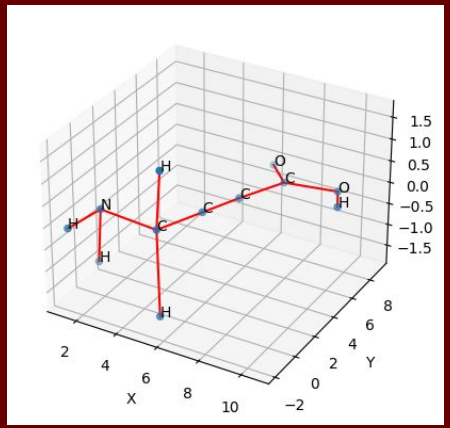
We see H and C are the most relevant, could be due to the fact that they mostly share single bonds, which are the most stable bonds, hence require much more energy to break them.

Contribution of each atom in Prediction:

	1.0	6.0	7.0	8.0	16.0	relevance
0	0	0	1	0	0	-0.360236
1	0	1	0	0	0	-0.638652
2	0	1	0	0	0	-0.638652
3	0	1	0	0	0	-0.638652
4	0	1	0	0	0	-0.638652
5	0	0	0	1	0	-0.330260
6	0	0	0	1	0	-0.330260
7	1	0	0	0	0	-0.785421
8	1	0	0	0	0	-0.785421
9	1	0	0	0	0	-0.785421
10	1	0	0	0	0	-0.785421
11	1	0	0	0	0	-0.785421
12	0	0	0	0	0	0.000000
13	0	0	0	0	0	0.000000
14	0	0	0	0	0	0.000000
15	0	0	0	0	0	0.000000
16	0	0	0	0	0	0.000000
17	0	0	0	0	0	0.000000
18	0	0	0	0	0	0.000000
19	0	0	0	0	0	0.000000
20	0	0	0	0	0	0.000000
21	0	0	0	0	0	0.000000
22	0	0	0	0	0	0.000000

However, considering the scatterplot it might have the following atomic structure:





So far, we do not account for relevance by type of bond, it just helps us understand the contribution by each type of atom.

3.1.

Simple atom-based Representation

Mapping each molecule to Vector and applying Regression

```
'''
As explained above, for each molecule, now we sum over one hot encoding of each atom to get uniform representation
'''
one_hot_encoded_array = np.array(onehot_encoding)
one_hot_df = pd.DataFrame(np.sum(one_hot_encoded_array, axis=1), columns=unique_atoms_list)
one_hot_df_array = one_hot_df.to_numpy()
one_hot_df_array.shape
one_hot_df_array[:5]
```

```
array([[4, 1, 0, 0, 0],
       [6, 2, 0, 0, 0],
       [4, 2, 0, 0, 0],
       [2, 2, 0, 0, 0],
       [6, 2, 0, 1, 0]])
```

From One Hot Encodings of each atom, we get their Vector representation by summing up the type of atoms. For example, the first molecule (CH4) is represented as {4,1,0,0,0} as shown above.

We then perform Ridge Regression and see contribution of each type of atom into predicting the atomization energy as shown before.

3.2. Models with Pairs of Atoms

Molecule Representation based on Pairwise Distance between atoms (Hard Indicator)

Building ϕ^A:

We represent each molecule based on the pairwise distances between each combination of atoms. Here is an example of the first molecule CH4:

```
distance_info[0]
```

```
[{'Molecule': 0, 'Atom Pair': 'C-H', 'Distance': 2.063549041748047},
 {'Molecule': 0, 'Atom Pair': 'C-H', 'Distance': 2.0635344982147217},
 {'Molecule': 0, 'Atom Pair': 'C-H', 'Distance': 2.0635828971862793},
 {'Molecule': 0, 'Atom Pair': 'C-H', 'Distance': 2.0651566982269287},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.37018084526062},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.3701982498168945},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.3706564903259277},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.370192527770996},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.3706531524658203},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.3706717491149902}]
```

We initially take the interval 0-1, 1-2, .., 5-6 of unit distance. ϕ^A for CH4 then looks like :

```
intervals = [(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6)]
results = create_one_hot_encoding(non_zero_coordinates, intervals)
```

```
results[0] #for 1st molecules with 5 atoms
```

```
array([[0, 0, 1, 0, 0, 0],
       [0, 0, 1, 0, 0, 0],
       [0, 0, 1, 0, 0, 0],
       [0, 0, 1, 0, 0, 0],
       [0, 0, 0, 1, 0, 0],
       [0, 0, 0, 1, 0, 0],
       [0, 0, 0, 1, 0, 0],

      [0, 0, 0, 1, 0, 0],
      [0, 0, 0, 1, 0, 0],
      [0, 0, 0, 1, 0, 0]])
```

Molecule Representation based on Pairwise Distance between atoms (Soft Indicator)

Building ϕ^A:

To avoid unnatural discontinuation, we replace the previous method by a Gaussian Function with mean at the centre of the interval with a fixed variance.

```
distance_info[0]
```

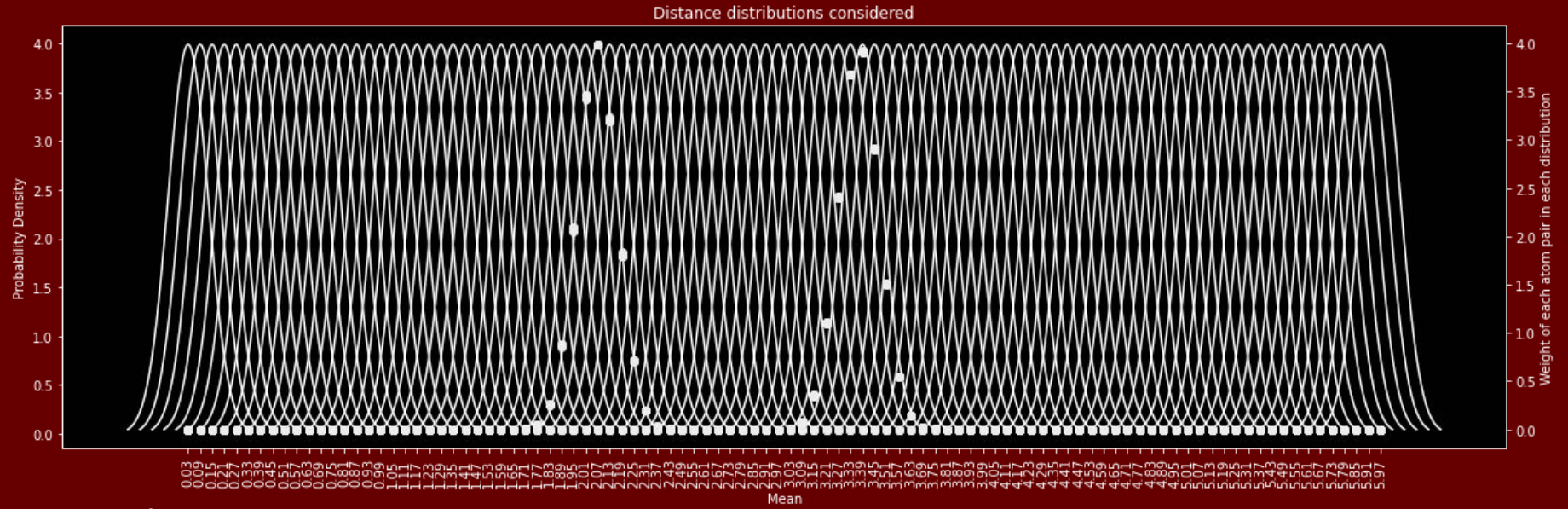
```
[{'Molecule': 0, 'Atom Pair': 'C-H', 'Distance': 2.063549041748047},
 {'Molecule': 0, 'Atom Pair': 'C-H', 'Distance': 2.0635344982147217},
 {'Molecule': 0, 'Atom Pair': 'C-H', 'Distance': 2.0635828971862793},
 {'Molecule': 0, 'Atom Pair': 'C-H', 'Distance': 2.0651566982269287},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.37018084526062},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.3701982498168945},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.3706564903259277},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.370192527770996},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.3706531524658203},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.3706717491149902}]
```

We make the intervals much more continuous, taking 100 intervals between 0 to 6 (each of length 0.06). We choose a variance of 0.01 for the Gaussian Distribution.

ϕ^A for CH4 now has the shape 10 * 100

Molecule Representation based on Pairwise Distance between atoms (Soft Indicator)

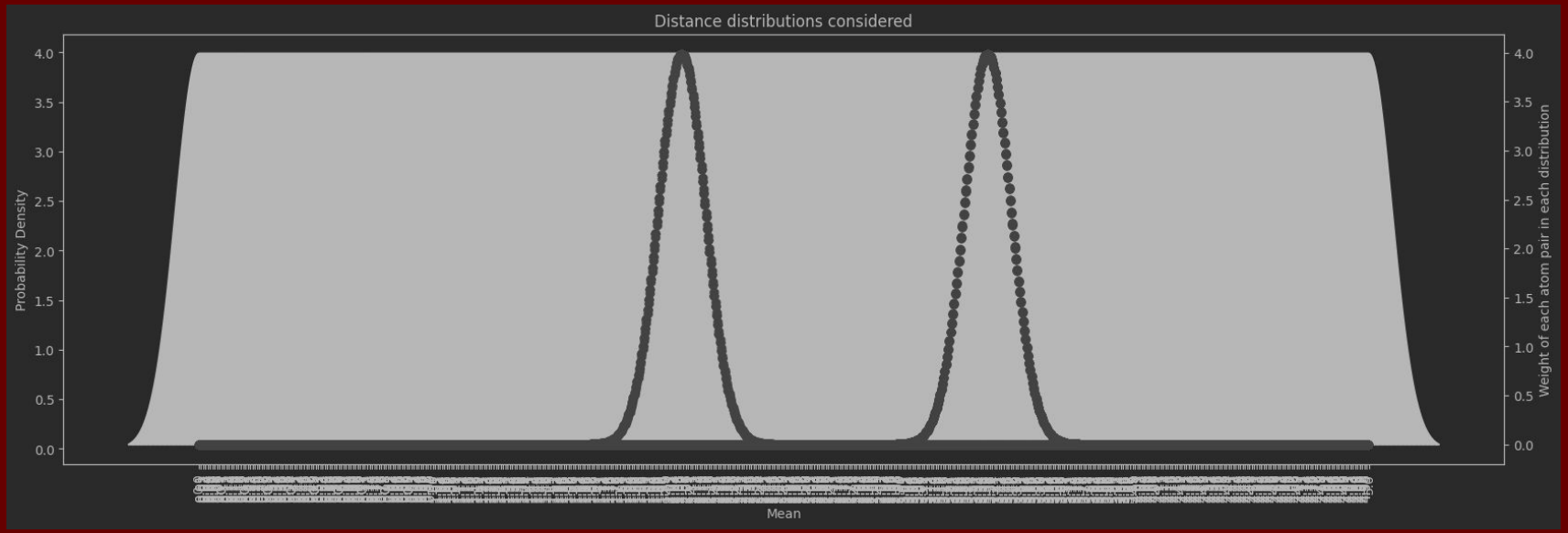
Visualizing ϕ^A for CH4:



With our soft encoding we have a peak on the distributions with mean 2.07 and 3.39, which seems to match the empirical data.

Molecule Representation based on Pairwise Distance between atoms (Soft Indicator)

We further experimented with different parameters.



Here, we chose the intervals much smaller and got more continuous soft indicator scores.

Molecule Representation based on Atom Bonds

Building ϕ^{B}:

We have five unique type of atoms, H, C, N, O and S. Therefore, there are 15 different type of unique bonds (unordered) between them:

```
['HH', 'HN', 'HO', 'HS', 'CH', 'CC', 'CN', 'CO', 'CS', 'NN', 'NO', 'NS', 'OO', 'OS', 'SS']
```

```
distance_info[0]
```

```
[{'Molecule': 0, 'Atom Pair': 'C-H', 'Distance': 2.063549041748047},
 {'Molecule': 0, 'Atom Pair': 'C-H', 'Distance': 2.0635344982147217},
 {'Molecule': 0, 'Atom Pair': 'C-H', 'Distance': 2.0635828971862793},
 {'Molecule': 0, 'Atom Pair': 'C-H', 'Distance': 2.0651566982269287},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.37018084526062},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.3701982498168945},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.3706564903259277},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.370192527770996},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.3706531524658203},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.3706717491149902}]
```

After trial and error, we decide if the pairwise distance between two atoms are less than 2.5 unit, they have a bond. For example, in CH4, there are four bonds between C and H but no bonds between the H atoms.

Molecule Representation based on Atom Bonds

Building ϕ^B:

(Example for CH4)

```
distance_info[0]
```

```
[{'Molecule': 0, 'Atom Pair': 'C-H', 'Distance': 2.063549041748047},
 {'Molecule': 0, 'Atom Pair': 'C-H', 'Distance': 2.0635344982147217},
 {'Molecule': 0, 'Atom Pair': 'C-H', 'Distance': 2.0635828971862793},
 {'Molecule': 0, 'Atom Pair': 'C-H', 'Distance': 2.0651566982269287},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.37018084526062},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.3701982498168945},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.3706564903259277},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.370192527770996},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.370653152465820 3},
 {'Molecule': 0, 'Atom Pair': 'H-H', 'Distance': 3.3706717491149902}]
```

```
   HH  HN  HO  HS  CH  CC  CN  CO  CS  NN  NO  NS  OO  OS  SS
[[0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
 [0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
 [0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
 [0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
 [0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
 [0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
 [0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
 [0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
 [0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
 [0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0]]
```

$\phi^A * \phi^B$

To improvise the regression model, now we merge both ϕ^A and ϕ^B

100 rows | 100 rows × 15 columns np.ndarray

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
23	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
24	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26	0.0	0.0	0.0	0.0	0.00020	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
27	0.0	0.0	0.0	0.0	0.00303	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
28	0.0	0.0	0.0	0.0	0.03038	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
29	0.0	0.0	0.0	0.0	0.21217	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	1.03385	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
31	0.0	0.0	0.0	0.0	3.51472	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
32	0.0	0.0	0.0	0.0	8.33657	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
33	0.0	0.0	0.0	0.0	13.79578	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
34	0.0	0.0	0.0	0.0	15.92819	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
35	0.0	0.0	0.0	0.0	12.83063	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
36	0.0	0.0	0.0	0.0	7.21092	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
37	0.0	0.0	0.0	0.0	2.82746	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
38	0.0	0.0	0.0	0.0	0.77351	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
39	0.0	0.0	0.0	0.0	0.14763	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
40	0.0	0.0	0.0	0.0	0.01966	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
41	0.0	0.0	0.0	0.0	0.00183	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
42	0.0	0.0	0.0	0.0	0.00012	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
43	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
44	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

The product feature map has shape of 100x15

- 100 is the dimension from ϕ^A - 100 intervals
- 15 is the dimension of ϕ^B - 15 distance atom bonds

Here is an example of CH4, where the feature values are high for (34,4) where 34th interval represents the closest bond distance in the molecule - CH

$\phi^A * \phi^B$ - Ridge Regression - again!

WIth the updated feature map of ϕ^A and ϕ^B we again run the ridge regression to model the atomization energy.

```
ridge_pair = Ridge(alpha=lambda_val,fit_intercept=False)
ridge_pair.fit(X_train_scaled, y_train_scaled)
```

Executed at 2023.07.12 15:05:55 in 371ms

```
Ridge
Ridge(alpha=0.01, fit_intercept=False)
```

```
print(f"Average error in predicted and actual atomization energy: {round((ridge_result_on_y_test_scaled - y_test_scaled).mean(),4)}")
```

Executed at 2023.07.12 20:13:07 in 64ms

```
Average error in predicted and actual atomization energy: 0.0165
```

Pairwise Potentials

The atomisation energy that we modelled now is a function of both atomic bond distance and the bond type.

```
1  X_one_molecule = flattened_X[0]
2  molecule_relevance_by_each_feature = ridge_pair_weights * np.array(X_one_molecule) #=w·phiA*phiB
3  molecule_relevance_by_each_feature = molecule_relevance_by_each_feature.reshape(phi_A_phi_B.shape)
4  molecule_relevance_by_each_feature.shape
```

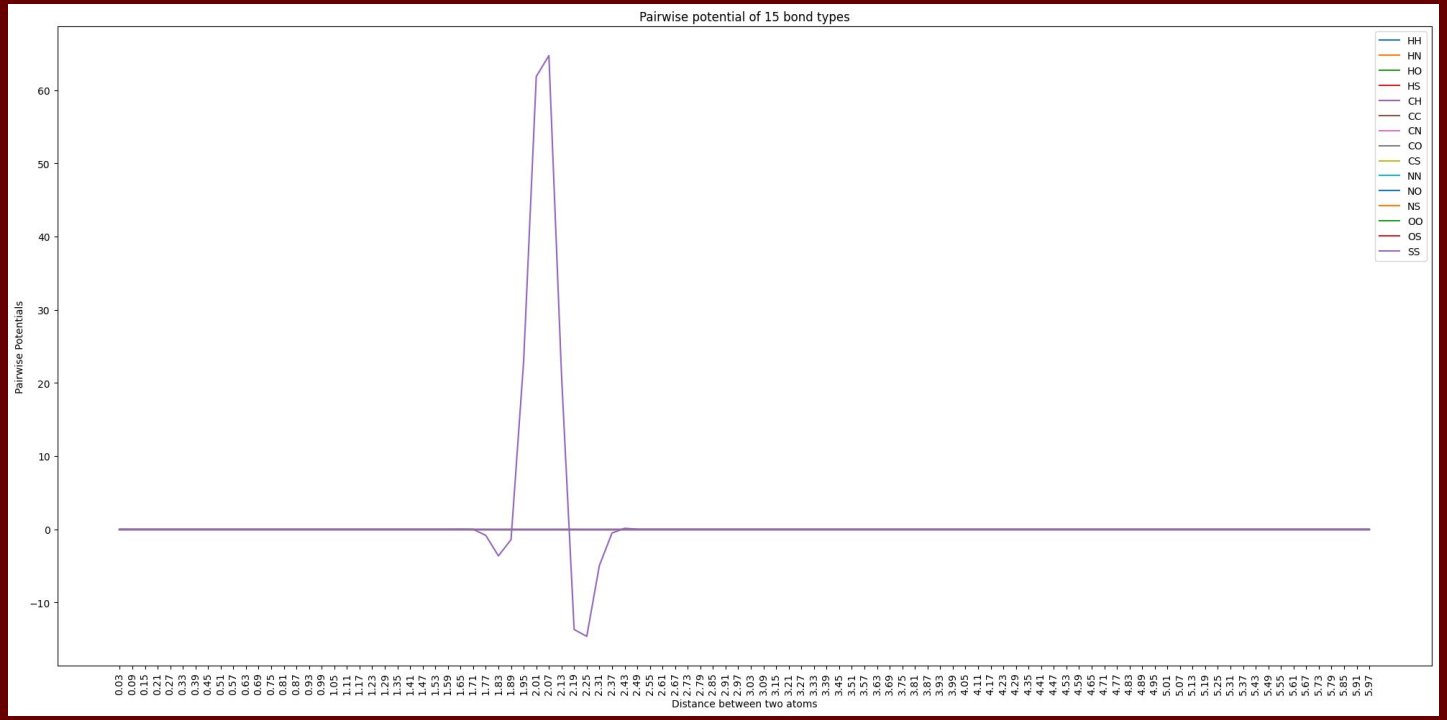
Executed at 2023.07.12 15:05:56 in 107ms

```
(100, 15)
```

To visualize it further, we calculate the relevance by each feature

- Multiply ridge regression weights/parameters with the atom representation
- Reshape the result to get a matrix where each value is a function of both atom bond distance and bond type

Pairwise Potentials



This is again for CH4 molecule, where the atomization energy required to break the bond is higher around the distance 2.06 units, which is roughly the actual distance between C and H.

If the distance is higher, it probably does not take much energy or releases energy for the bond to break.