Predicting Molecular Properties

Anmol Srivastava[1]**, Saiesh Kumar Jain**[2]
Khoury College of Computer Sciences
Northeastern University
360 Huntington Avenue, Boston, MA 02115
srivastava.anm@northeastern.edu[1] jain.sai@northeastern.edu[2]

Abstract

Medical Researchers around the world exploit the properties of the molecules to understand its structure and chemical behavior which helps them to design molecules to carry out specific task and produce new drugs to fight deadly disease. The knowledge of magnetic interactions between the atoms of molecule i.e. Scalar Coupling Constant helps them to infer various properties of molecules. However, calculating these interaction via tradition quantum mechanics methods is time consuming and computational extensive, which opens up the research for other predicting methodologies like statistical machine learning and deep learning algorithms. In addition, the domain of variables involves could be narrowed down via feature selection algorithms and transfer learning.

Nuclear Magnetic Resonance (NMR) spectroscopy is an analytical chemistry technique used in quality control and research for determining the content and purity of a sample as well as its molecular structure. This technique could be used to gain insight into a molecule's structure and dynamics depends on the ability to accurately predict so-called "scalar couplings". These are the magnetic interactions between the pair of atoms whose strength depends upon intervening electrons and chemical bonds that make up a molecule's three-dimensional structure.

Predicting these interactions accurately will allow medicinal chemists as well as researchers to gain deeper insights into the molecular structures which would facilitate their understanding of molecule's properties and behavior on the basis of its three-dimensional chemical structure which in turn could provide a solution to important problems like designing molecules to carry out specific cellular tasks, or designing better drug molecules to fight disease.

Methods of quantum mechanics are the traditional approach for this task. However, the calculations involved are computationally expensive and time-consuming. Therefore, resorting to Machine Learning and Deep Learning could prove to be less computationally expensive and time-consuming as compared to the traditional approaches.
In this project, we are going to write algorithms to predict the interaction between two atoms in a molecule using different Machine Learning and Deep Learning techniques. After fitting the models and neural network to our data set, we will look for subset of features which could do the job with somewhat close accuracy to that of original features. This would obscure the need of default 37 features.

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Dataset

The entire data is dispersed in seven csv files and consists of total 4.7 million entries collected by the means of NMR experiments. This dataset then could be used for training deep learning and machine learning models after preprocessing it. The respective csv files consists of following features of molecules:

- **Default Data** consists of atom-pair type and indexes, coupling type and our target variable scalar coupling constant.
- **dipolemoments.csv** contains the molecular electric dipole moments which indicates the charge distribution in the molecules.
- **structures.csv** has information about the origin of coupling constant in the molecule.
- **magneticshieldingtensors.csv** contains the magnetic shielding tensors for all atoms in the molecules.
- **mullikencharges.csv** contains the mulliken charges for all atoms in the molecules.
- **potentialenergy.csv** contains the potential energy of the molecules.
- **scalarcouplingcontributions.csv** consists of various types of coupling constants in the molecule.

Project Outline

The first stage is **Data Preprocessing and Data Wrangling** in which the dispersed data is combined into single data frame which undergoes preprocessing phase to eliminate redundant information and convert categorical encoding of some features to numeric encoding.

Machine Learning stage involves using the preprocessed data to train regression machine learning models in order to predict our target variable. In addition,, feature selection technique have also been used to select 10 most prominent features out of 37 default features that contributes towards the objective of this project.

Deep Learning phase involves implementing various deep neural networks architecture to predict the magnetic interactions between the atoms pair of the molecule. Optimization of these neural nets would be also done via hyperparameter tuning which culminates with search of subsets of feature by the means of transfer learning.

Consequentially, at the end all trained models' performance would be compared to find the optimal algorithms for this problem.

Data Wrangling and Preprocessing

Final dataset is formed after combining the data from all six csv files using the following wrangling algorithm.

Algorithm:

```
function dataset_creator(left_df,right_path,
                        left_key,right_key,
                        rename_dict,drop_list)

    right_df <-- load_dataset(right_path)
    for i in length(left_key):
        train_df <-- pd.merge(left_df,right_df,
                              left_key[i],right_key)

    train_df <-- pd.rename(train_df,rename_dict)
    train_df <-- pd.drop(train_df,drop_list)

returns a modified dataframe
```

This algorithm accepts the two datasets which are to be merged along with primary keys for both the datasets and merge them on the basis of common key values between them. Moreover, some redundant columns could also be removed or renamed by assigning their names to *drop_list* and *rename_dict* respectively.

Machine Learning Algorithms are based on mathematical concepts and work on numerical variables. Some of the features in dataset are of categorical type which needs to be converted into numeric encoding.

atom_0 variable specify the type of atom 0 and its domain value is **"H"** which gets converted to integer 0. Similarly, **atom_1's** domain values 'C','H', and 'N' gets encoded to 0,1 and 2 respectively. In case of **type** variable which illustrate the type of coupling between the atoms of molecules also undergoes categorical encoding process with some modification. The domain of this variables consists of eight unique values, **'1JHC', '2JHH', '1JHN', '2JHN', '2JHC', '3JHH', '3JHC', '3JHN'**. It's domain could be narrowed down to '1J', '2J', '3J' because the information about the atom pair present in the variable is already being depicted with variables *atom_0, atom_1*. Therefore, the only unique feature depicted by this variable is type of couplings i.e. '1J', '2J', and '3J'. Lastly, all these numeric encoding gets converted into dummy variables by the means of one-hot encoding.

The final pre-processed data with 37 features and 4.7 million entries is split into the training set and test set with the splitting ratio of 8:2 which means 80% of the total data is present in training set i.e. **3727260** samples and 20% of our data in present in the test set i.e. **465908** samples, on which our model will be trained and tested respectively.

Machine Learning

Machine learning is an application of *artificial intelligence (AI)* that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide**(de MelloMoacir Antonelli Ponti 2018)**. The primary aim is to allow the Models learn automatically without human intervention or assistance and adjust actions accordingly. Machine learning enables analysis of massive quantities of data, that's why it was our first choice for predicting scalar coupling constant. Machine learning implementations are generally divided into 3 categories depending on the nature of the learning "signal" or "response" available to a learning system which are Supervised Machine learning algorithms, Unsupervised Machine learning algorithms, Reinforcement learning algorithms. Another categorization of machine learning tasks arises when one considers the desired output of a machine-learned system: Classification, Regression, Clustering. As our target variable *Scalar coupling constant* is a real valued number and also we had our example data with associated target responses, we have used Regression Machine learning Algorithms such as Linear Regression, Ridge Regression, Lasso Regression.
Having a good understanding of feature selection/ranking methods leads to better performing models, better understanding of the underlying structure and characteristics of the data and leads to better intuition about the algorithms that underlie many machine learning models. There are in general two reasons why feature selection is used:
1. Reducing the number of features, to reduce over-fitting and improve the generalization of models.
2. To gain a better understanding of the features and their relationship to the response variables.
So for the same we have plotted heat map and performed Pearson co-relation variable selection method.

Supervised Machine Learning Regression Algorithms

Supervised machine learning algorithms can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training data set, the learning algorithm produces an inferred function to make predictions about the output values. The model is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.

Metrics for Model analysis and evaluation

There are many metrics to evaluate and analyze machine learning algorithm. We have used following mentioned metrics to compare our different machine learning models and do regression analysis.

1. **Coefficient of determination (R-Squared value) :-**
   R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression. The definition of R-squared is fairly straight-forward; it is the percentage of the response variable variation that is explained by a linear model.

R-squared = Explained variation / Total variation

2. **Mean absolute error :-**
Mean Absolute Error is a model evaluation metric used with regression models. The mean absolute error of a model with respect to a test set is the mean of the absolute values of the individual prediction errors on over all instances in the test set. Each prediction error is the difference between the true value and the predicted value for the instance

$mae = \sum_{i=1}^{n} abs\,(y_i - (x_i))$/n

3. **Mean squared error :-**
Mean Squared Error is a model evaluation metric often used with regression models. The mean squared error of a model with respect to a test set is the mean of the squared prediction errors over all instances in the test set. The prediction error is the difference between the true value and the predicted value for an instance.

$mse = \sum_{i=1}^{n} (y_i - (x_i))^2$/n

4. **Root mean squared error :-**
Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit.

$mse = \sqrt{\sum_{i=1}^{n} (y_i - (x_i))^2 / n}$

Linear Regression

Linear Regression is a linear model, i.e. a model that assumes linear relationship between the input variables(x) and the single output variable (y). It predicts a dependent variable value (y) based on a given independent variables (x1,x2… , xn). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression(**analyticsvidhya April 2020**).

$$y = m + a_1x_1 + a_2x_2 + ...a_nx_n$$

Equation 1 : *Linear Model*

Here our model looks for optimizing the values of intercept and independent coefficients value such that it minimizes the cost function given by equation using gradient descent optimizing algorithm.

$$\sum_{n=1}^{M} (y_i - \hat{y}_i)^2 = \sum_{n=1}^{M} (y_i - \sum_{j=0}^{p} a_j * x_{ij})^2$$

Equation 2 : *M :- No. of instance, p :- No. of features*

We have used scikit-learn library to perform linear regression. As 80 % is our training data, we will generalise the rest 20 % of data according to training of above 80 % of data, so that we can overcome the problem of overfitting and can predict our target variable *Scalar coupling constant*. The Algorithm for Linear regression is as follow:

1. Load the data and split it in 80 : 20 ratio.
2. Visualize the data.
3. Calculate coefficients (slope and intercepts)
4. Calculate the predicted value
5. Plot and analyze predicted v/s true value.

Performance: To analyze the performance of Linear Regression on test set we have made use of R-square and Mean Square Error matrices. In addition, graphs have also been used to visualize it.

Table 1: *Performance Matrices*

R-Square	MAE	MSE	RMSE
0.999	2.27e^{-05}	6.68e^{-09}	8.17e^{-05}

From ***Table 1***, it can be inferred that Linear Regression performs exceptionally well with 37 input features as the r-square is very close to 1 with low MSE.

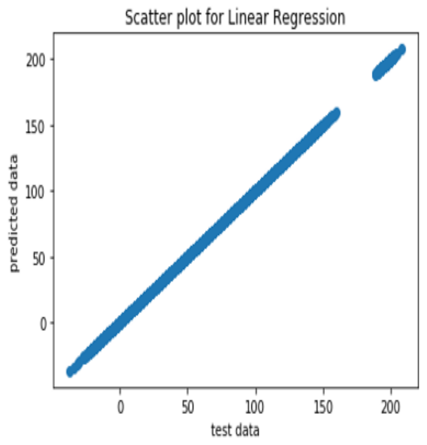

Figure 1: *predicted data v/s test data*

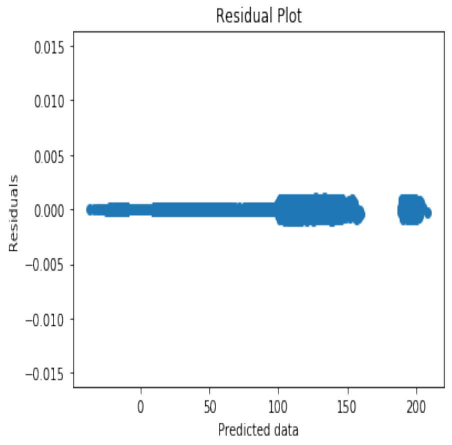

Figure 2: *Residuals v/s Predicted data*

Inference's It is evident from Scatter plot and Residual plot that, that the data has a linear pattern and it would not be viable to do perform non-linear model regression. But As we have million rows in data-set and number of features are also very high, therefore using regularization regression techniques such as Ridge and Lasso Regression would be good which reduce the complexity of regression models by putting pressure on the absolute size of the coefficients, driving some to zero.

Ridge Regression

Ridge regression is a technique for analyzing multiple regression data. When multicollinearity occurs, least squares estimates are unbiased. A degree of bias is added to the regression estimates, and a result, ridge regression reduces the standard errors(**Springer 2008**). It is done by penalizing high valued regression coefficients.Ridge uses L2 regularization i.e it adds a L2 penalty equal to the square of the magnitude of coefficients. L2 will not yield sparse models and all coefficients are shrunk by the same factor (none are eliminated)(**Bhattacharyya April 2020**).

$$\sum_{n=1}^{M}(y_i-\hat{y}_i)^2=\sum_{n=1}^{M}(y_i-\sum_{j=0}^{p}a_j*x_{ij})^2+\alpha\sum_{j=0}^{p}a_j^2$$

Equation 3 :- *Cost function for Ridge Regression*

We have scikit-learn library to train our model using ridge regression. we have iterated our model through different α value and recorded metrics for analyzing the model. For $\alpha = 0.05$ we got our scatter plot as shown below and the R^2 value for this is near about 1, thus good for generalising the new data.

Performance: To analyze the performance of Ridge Regression on test set we have made use of R-square and Mean Square Error matrices. In addition, scatter plot have also been used to visualize it.

Table 2: *Performance Matrices*

R-Square	MAE	MSE	RMSE
0.999	0.00018	$8.84e^{-08}$	$8.17e^{-08}$

From ***Table 2***, it can be inferred that ridge Regression performs exceptionally well with 37 input features as the r-square is very close to 1 with low MSE.

Inference's Ridge performs well the model by penalizing the coefficient of independent variables. we can also conclude that as we increase the value of α, complexity of model decreases but the R-square value goes on decreasing and there was over-fitting of data as we there was a difference between R-Square for train dataset and test data-set.

Lasso Regression

LASSO stands for "Least Absolute Shrinkage and Selection Operator". There are two main keywords here i.e. 'absolute' and 'selection'. Lasso regression performs L1 regularization, i.e. it adds a factor of sum of absolute value of coefficients in the optimization objective which is key difference between Ridge and Lasso. Also Lasso do variable selection itself i.e. penalizing and eliminating some of the independent variables.

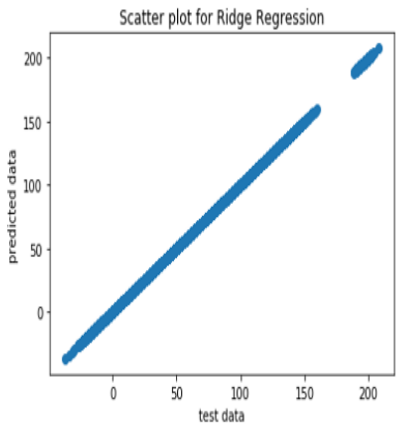

Figure 3: *Residuals v/s Predicted data*

$$\sum_{n=1}^{M}(y_i-\hat{y}_i)^2=\sum_{n=1}^{M}(y_i-\sum_{j=0}^{p}a_j*x_{ij})^2+\alpha\sum_{j=0}^{p}|a_j|$$

Equation 4 :- *Cost function for Lasso Regression*

Performance: To analyze the performance of Lasso Regression on test set we have made use of R-square and Mean Square Error matrices. In addition, graphs have also been used to visualize it.

Table 3: *Performance Matrices*

R-Square	MAE	MSE	RMSE
0.999	0.0.0698	0.00944	0.00947

From ***Table 3***, it can be inferred that Lasso Regression performs exceptionally well with 11 input features as the r-square is very close to 1 with low MSE.

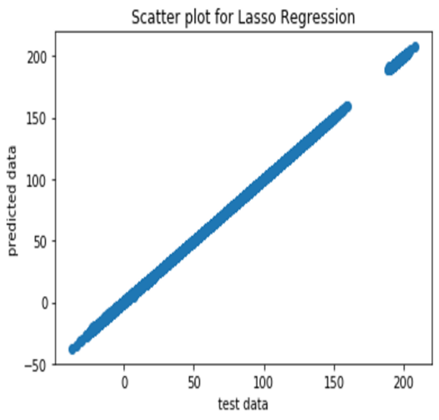

Figure 4: *Residuals v/s Predicted data*

Inference's As hyper parameter alpha increases the bias for each independent variable is increased and more more of them are eliminated, but at the same over-fitting also increases. for alpha = 0.01 and for 10^5 iteration the R-Square value was high, and all error metrics were low, and thus the value of scalar coupling constant is predicted correctly. We were able to narrow down number of features to 10 and we verified the same by using Pearson coefficient variable selection method.

Pearson coefficient variable selection method

Pearson correlation coefficient, which measures linear correlation between two variables. The resulting value lies in [-1;1], with -1 meaning perfect negative correlation (as one variable increases, the other decreases), +1 meaning perfect positive correlation and 0 meaning no linear correlation between the two variables(**datadive April 2020**). This is a filter-based method. We check the absolute value of the Pearson's correlation between the target and numerical features in our dataset. We keep the top n features based on this criterion.

Table 4: *Pearson Co-filter Features*

atom_index_1	YYmag_1	XXmag_1
ZZmag_1	atom1_0	sd
mulliken_charges_1	Dso	type
atom_index_0		

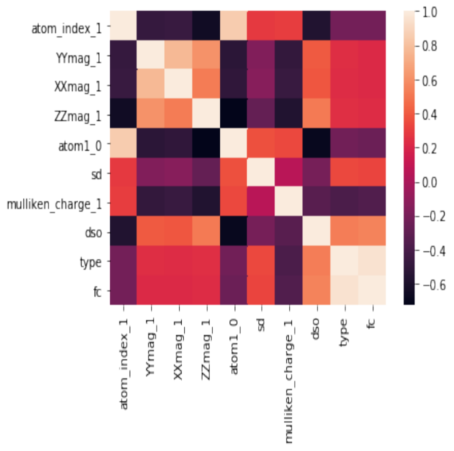

Figure 5: *Heatmap for selected predictors*

Performance: To analyze the performance of Linear Regression on test set with 10 significant features we have made use of R-square and Mean Square Error matrices. In addition, graphs have also been used to visualize it.

Table 5: *Performance Matrices*

R-Square	**MAE**	**MSE**	**RMSE**
0.999	0.0132	0.06999	0.2645638

From ***Table 4***, it can be inferred that Linear Regression performs exceptionally well with 10 input features as the r-square is very close to 1 with low MSE.

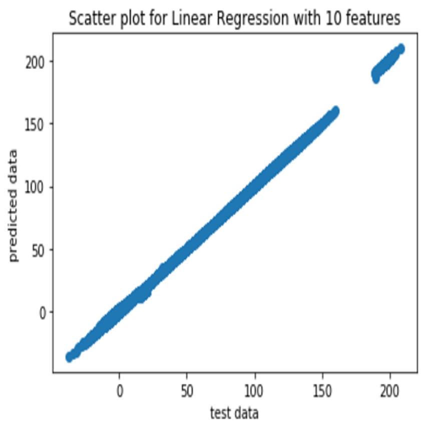

Figure 6: *predicted data v/s test data*

Inference's As choosing 10 custom features, computational time was decreased by a great deal and at the same time, R-square value was near to 1 and all error metrics were low. The same was proved in transfer learning phase of our model.

Deep Learning

Deep learning is a specific sub-field of machine learning: a new take on learning representations from data that puts an emphasis on learning successive layers of increasingly meaningful representations. The number of layers that contribute to the model of a data is known as *depth* of a model. The models of deep learning are known as *neural networks* which consists of stack of different layers on top of each other **(Chollet 2017)**.

Working of Artificial Neural Networks

In order to solve the regression problem of predicting **Scalar Coupling Constant** on the basis of 37 independent variables, we will be using several neural network architecture with different number of neural layers. The connection between the each neural layer in the network is represented by the weight matrix, $'W^{[l]}' \in \mathbb{R}^{\text{no. of units in next layer x no. of units in the previous layer}}$.

$$W^{[l]} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,j} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ w_{i,1} & w_{i,2} & \cdots & w_{i,j} \end{bmatrix}$$

Xavier Initialization: These weight matrices are initially initialized by Xavier initialization whose aim is to prevent the vanishing or exploding gradient problem during the phase of forward propagation.**(Xavier Glorot 2010)** It initializes the weight of each layer by sampling it from uniform distribution of **[-r, +r]** where,

$$r = \frac{\sqrt{2}}{\sqrt{n_{\text{in}} + n_{\text{out}}}}$$

n_{in}: no. of input neurons in weight tensor, n_{out}: no. of output neurons in weight tensor

Activation Function: The activation function used between the hidden layer is ReLU activation function which is a piece wise linear activation that outputs the input directly if it is positive else output zero.**(Chigozie Enyinna Nwankpa 2018)** This is given by the mathematical formula $ReLU(Z) = max(z, 0)$

$$Z = W^{T}X + b$$
$$a = g^{[l]}(Z)$$

where $g^{[l]}$ denotes the l^{th} layer activation function.

Optimizer: All the neural networks developed for this project makes use of Adam Optimizer which is a combination of RMSprop and Stochastic Gradient Descent with momentum. It uses the squared gradients to scale the learning rate like RMSprop and it takes advantage of momentum by using moving average of the gradient.**(Diederik P. Kingma 2015)**

Algorithm

$$V_{dw} = 0, S_{dw} = 0, V_{db} = 0, S_{db} = 0$$

for iteration t:

$$V_{dw} = \beta_1 V_{dw} + (1-\beta_1)dw, V_{db} = \beta_1 V_{db} + (1-\beta_1)db$$
$$S_{dw} = \beta_2 S_{dw} + (1-\beta_2)dw, S_{db} = \beta_2 S_{db} + (1-\beta_2)db$$
$$V^{corr}{}_{dw} = \frac{V_{dw}}{(1-\beta^{t}{}_1)}, S^{corr}{}_{dw} = \frac{S_{dw}}{(1-\beta^{t}{}_2)}$$
$$V^{corr}{}_{db} = \frac{V_{db}}{(1-\beta^{t}{}_1)}, S^{corr}{}_{db} = \frac{S_{db}}{(1-\beta^{t}{}_2)}$$
$$W := W - \alpha\frac{V^{corr}{}_{dw}}{\sqrt{S^{corr}{}_{dw}} + \epsilon}, b := b - \alpha\frac{V^{corr}{}_{db}}{\sqrt{S^{corr}{}_{db}} + \epsilon}$$

MolNet Basic Neural Network

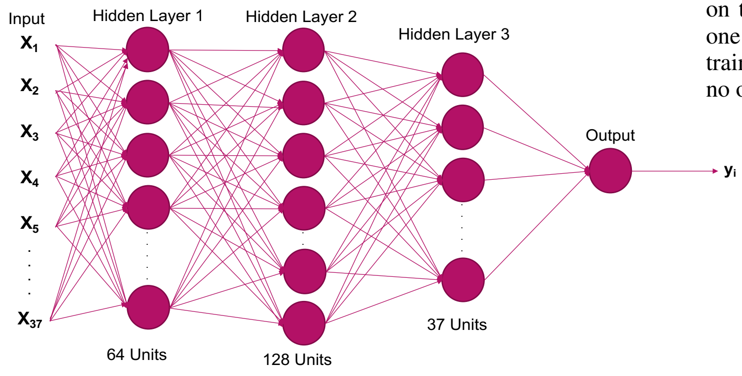

Figure 7: *MolNet Basic Architecture with three hidden layers*

The first neural network used for predicting magnetic interactions between the atoms is artificial neural network named as MolNet. ***Figure 7*** shows the architecture which is a stack of three hidden layers. First, Second and Third hidden layers has 64, 128 and 37 neuron units respectively. ReLU activation is used between all these hidden layers with linear activation for output layer as the target variable is a continuous random variable. MolNet undergoes training on train set with following hyper-parameters.

Table 6: *Hyper-Parameters for MolNet Basic*

Learning Rate	Epochs	Optimizer	Dropout_rate
0.01	40	Adam	1.0

Callbacks techniques like **EarlyStopping** and **ReduceLROnPlateau** have been deployed to stop the training in case there is no reduction in validation loss or the training gets stuck on plateau.

Table 7: *Forward Propagation Output for MolNet Basic*

	Shape of W	Shape of b	Activation
Layer 1	(64, 37)	(64, 1)	$Z^{[1]} = W^{[1]}X + b^{[1]}$
Layer 2	(128, 64)	(128, 1)	$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$
Layer 3	(37, 128)	(37, 1)	$Z^{[3]} = W^{[3]}A^{[2]} + b^{[3]}$
Output	(1, 37)	(1, 1)	$Z^{[4]} = W^{[4]}A^{[3]} + b^{[4]}$

Table 8: *Performance Matrix for MolNet Basic*

R-Square	MSE
0.999	$1.337e^{-05}$

Performance: To analyze the performance of MolNet basic on test set we have made use of R-square and Mean Square Error matrices. In addition, graphs have also been used to visualize it.

From ***Table 8***, it can be inferred that MolNet performs exceptionally well with 37 input features as the r-square is very close to 1 with low MSE.

The ***Figure 8*** shows the no. of epochs on the *x-axis* & error on the *y-axis*. The orange line shows the training error and blue one shows the test error. As it can be seen that the test error and training error are close to each other which stipulates that there is no over-fitting to our data.

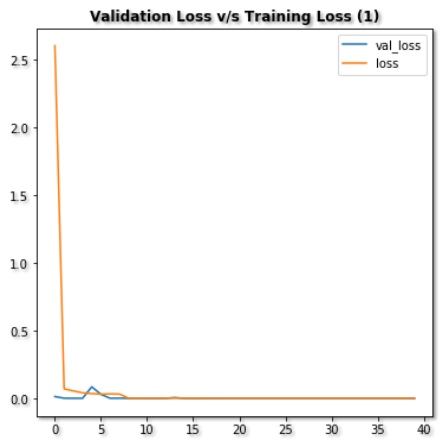

Figure 8: *Training & Test Error for MolNet Basic*

Scatter plot between the predicted values and true values as shown in ***Figure 9*** further corroborate the performance of our neural network as the correlation between these two quantities is almost linear without any noise.

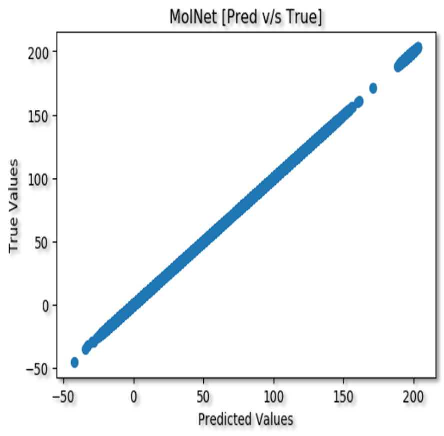

Figure 9: *True Values v/s Predicted Values for MolNet Basic*

MolNet Advance Architecture

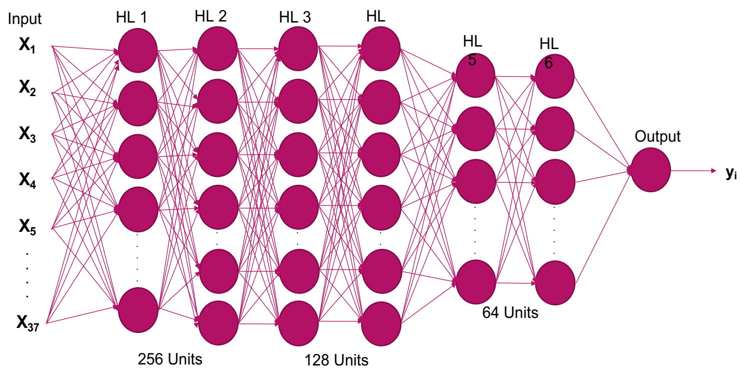

Figure 10: *MolNet Advance Architecture with six hidden layers*

The advance version of MolNet architecture consists of six hidden layer as shown in ***Figure 10*** where first two hidden layer has 256 neuron units. Similarly, hidden layer 3 & 4 is formed of 128 units, layer 5 & 6 has 64 units. It is trained on the train set with the following hyper-parameters:

Table 9: *Hyper-Parameters for MolNet Advance*

Learning Rate	**Epochs**	**Optimizer**	**Dropout_rate**
0.001	80	Adam	0.5

Table 10: *Forward Propagation Output for MolNet Advance*

	Shape of W	**Shape of b**	**Activation**
Layer 1	(256, 37)	(256, 1)	$Z^{[1]} = W^{[1]}X + b^{[1]}$
Layer 2	(256, 256)	(256, 1)	$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$
Layer 3	(128, 256)	(128, 1)	$Z^{[3]} = W^{[3]}A^{[2]} + b^{[3]}$
Layer 4	(128, 128)	(128, 1)	$Z^{[4]} = W^{[4]}A^{[3]} + b^{[4]}$
Layer 5	(64, 128)	(64, 1)	$Z^{[5]} = W^{[5]}A^{[4]} + b^{[5]}$
Layer 6	(64, 64)	(64, 1)	$Z^{[6]} = W^{[6]}A^{[5]} + b^{[6]}$
Output	(1, 64)	(1, 1)	$Z^{[7]} = W^{[7]}A^{[6]} + b^{[7]}$

Table 11: *Performance Matrix for MolNet Advance*

R-Square	**MSE**
0.961	47.454

Performance: The dropout rate of 0.5 is used to avoid overfitting which switches off the random neurons from neural layer with a probability of 50% during the training phase. However the ***Figure 11*** shows that the dropout doesn't help in this case. Moreover, it has worsen the performance of our network as test error is way to high than the training error.

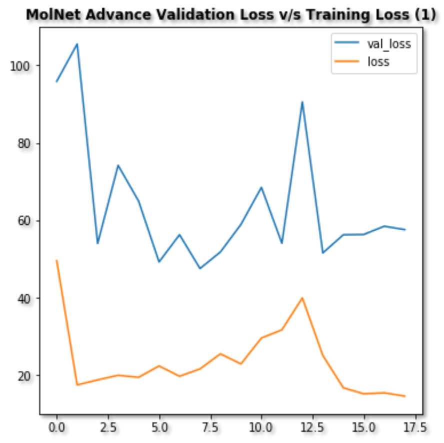

Figure 11: *Training & Test Error for MolNet Advance*

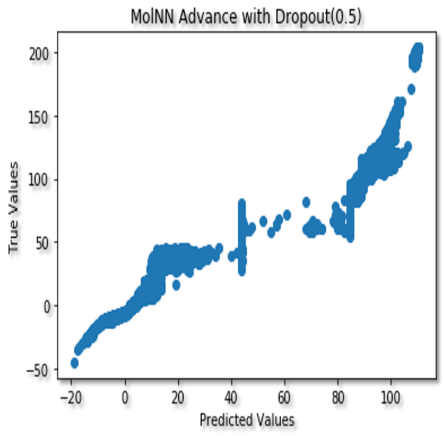

Figure 12: *True Values v/s Predicted Values for MolNet Advance*

Correlation between the predicted and the true value is also very noisy as seen from the ***Figure 12***.

MolNet Advance V1

This the version 1 of the previous MolNet Advance model with same neural architecture and hyper-parameters. However, this time it is trained on train data with the dropout rate of 1.0 which means no neurons will be dropped out from the layers during the training phase.

Performance: Above changes in the specifications shows the significant boost in the performance of model. R-Square is observed close to one with very low MSE. Moreover, the graphs also depicts the absence of over-fitting along with good correlation between the true and predicted quantity.

Table 12: *Performance Matrix for MolNet Advance V1*

R-Square	MSE
0.999	9.188e^{-07}

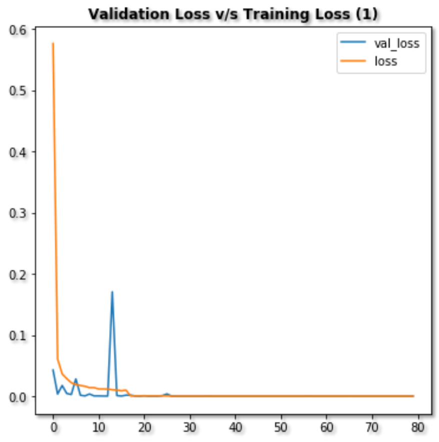

Figure 13: *Training & Test Error for MolNet Advance V1*

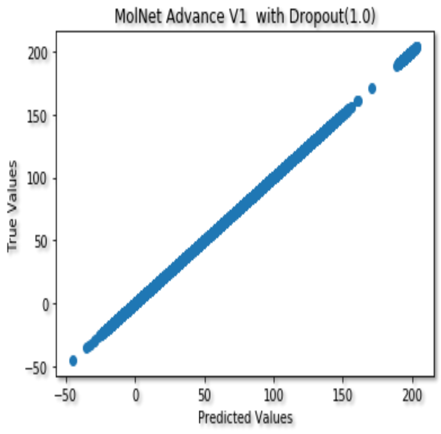

Figure 14: *True Values v/s Predicted Values for MolNet Advance V1*

Transfer Learning

Transfer Learning is the technique which allows us to leverage the previous learning on a particular domain and use it on different one without training the entire model from scratch. It is done by freezing the later layers of the model whose weights won't be updated during the back-propagation phase of learning. This is done because the depth patters are usually detected by the later layers rather than the former ones. Entire process will fine-tune the pre-trained neural network on the new domain.

Geometric Features:

We have noticed that with all 37 features our both basic and advance MolNet networks are able to achieve commendable performance. However, collecting this much of information is not always feasible as NMR computation methods are time consuming and expensive. Therefore, it opens up the research of selecting the most promising features which contributes to the success of model. **Geometric Structural Features** are the physical and structural properties of the molecule which are illustrated by the following ***atom_0,atom_index1,type,atom_0,atom1_0,atom1_1***

MolNet Basic V1

MolNet basic model is re-trained using six geometric features to predict the target variable by freezing third hidden layer during training phase. Although there is no over-fitting to data, the results that we got is not promising as the R-Square is low followed by high mean square error. Moreover, the correlation between the predicted and true values is also very noisy which suggests that our pre-trained model doesn't work with the new subset of features.

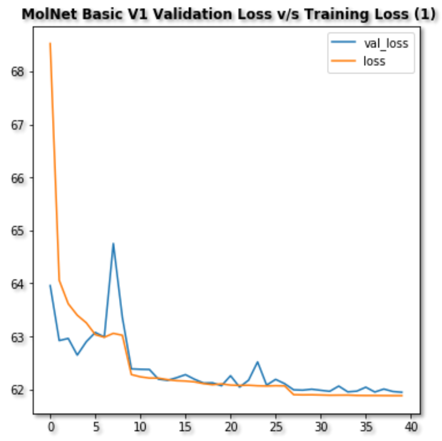

Figure 15: *Training & Test Error for MolNet Basic V1*

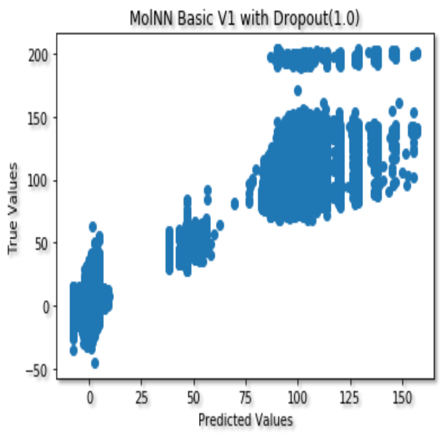

Figure 16: *True Values v/s Predicted Values for MolNet Basic V1*

MolNet Advance V1.1

MolNet V1.1 is based on advance MolNet architecture with the same hyper-parameter value as described in ***Table 9***. However, it is trained with only geometric structural features by freezing hidden layer 5th & 6th. The results were similar to that of Basic V1 i.e. no over-fitting with performance matrix stipulated by ***Table 14***.

Table 13: *Result: Deep Learning Phase*

Model Name	Inputs	Transfer Learning	Learning Rate	Epochs	R-Square	MSE	Dropout
MolNet Basic	37	No	0.01	40	0.999	1.33e^{-05}	1.0
MolNet Advance	37	No	0.001	80	0.961	47.45	0.5
MolNet Advance V1	37	No	0.001	80	0.999	9.18e^{-07}	1.0
MolNet Basic V1	6	Yes	0.01	40	0.949	61.82	1.0
MolNet Advance V1.1	6	Yes	0.001	80	0.953	62.57	1.0
MolNet Advance V1.1	12	Yes	0.001	80	0.991	10.16	1.0

Table 14: *Performance Matrix for MolNet Basic V1*

R-Square	MSE
0.949	61.82

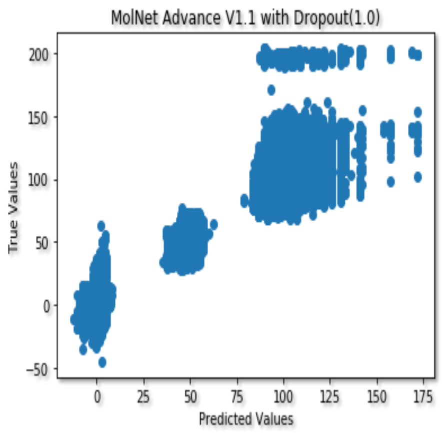

Figure 17: *True Values v/s Predicted Values for MolNet Advance V1.1*

Custom Pearson Features

Transfer Learning using only Geometric Structural Features is abortive in predicting the target variable. This suggest only physical features are not enough to fulfill our task. Therefore, our next approach would be to combine geometric features with some chemical features selected using the Pearson Co-filter approach in Machine Learning section of this paper which gives us twelve Custom Pearson Features.

Table 15: *Custom Pearson Features*

atom_index_1	YYmag_1	XXmag_1
ZZmag_1	atom1_0	sd
mulliken_charges_1	Dso	type
atom_index_0	atom1_1	atom_0

MolNet Advance V1.1 is retrained following the same training and hyper-parameters with Custom Pearson Features rather than geometric features. As a result, we get very good performance as this time our R-square is high followed by low mean_squared_error as shown in ***Table 16***. Also, the co-relation between the predicted and true values is less noisy this time.

Table 16: *Performance Matrix using Custom Pearson Features*

R-Square	MSE
0.991	10.162

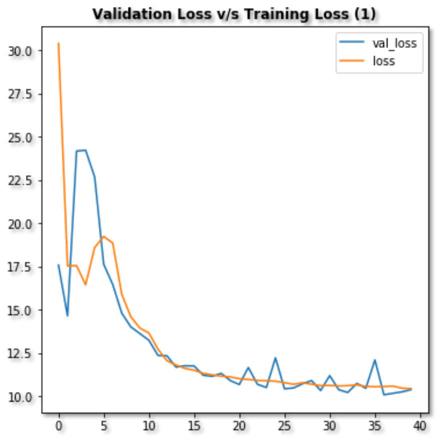

Figure 18: *Training & Test Error with Custom Pearson Features*

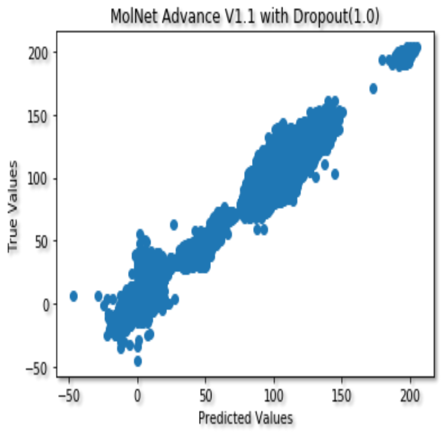

Figure 19: *True Values v/s Predicted Values with Custom Pearson Features*

Result & Conclusion

It is now evident that the problem of predicting Scalar Coupling Constant between the pair of atoms in the molecule can be predicted within a fraction of second with less computational power as compared to the traditional quantum mechanics methods by the means of machine learning and deep learning algorithms.

Since the collection of data is expensive for this job, we have searched for promising subset of features which could give us the approximate performance as we could get with conventional set of feature. Subsequently the assortment of geometric structural feature and pearson co-filter features known as Custom Pearson Features**[CPF]** is convenient to accumulate.

The results of Deep Learning Phase has been summarized in ***Table 13*** which shows that the MolNet Advance architecture with no dropout performs exceptionally well in calculating target variable with minimal error. However, when these neural architecture has been trained using six physical features of the atom they fall short in achieving good results. This discrepancy has been then successfully addressed by facilitating the training with CPF.

References

analyticsvidhya. April 2020. A comprehensive beginners guide for linear, ridge and lasso regression. https://www.analyticsvidhya.com/blog/2017/06/a-comprehensive-guide-for-linear-ridge-and-lasso-regression/.

Bhattacharyya, S. April 2020. L1 and l2 regularization. https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b.

Chigozie Enyinna Nwankpa, Winifred Ijomah, A. G. S. M. 2018. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv:1811.03378*.

Chollet, F. 2017. *Deep Learning with Python*. Manning Publications.

datadive. April 2020. Feature selection. https://blog.datadive.net/selecting-good-features-part-i-univariate-selection/.

de MelloMoacir Antonelli Ponti, R. F. 2018. *A Brief Review on Machine Learning*. Springer, Cham.

Diederik P. Kingma, J. L. B. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Springer. 2008. *Ridge Regression*. New York, NY: Springer New York.

Xavier Glorot, Y. B. 2010. Understanding the difficulty of training deep forward neural network. *International Conference on Artificial Intelligence and Statistics*.