ALY6020 Final Project : Prediction of Molecular Properties

Shivani Grover, Anish Nitin Somaiah

June 28, 2019

INTRODUCTION:

[1]There are two distinct types of magnetic interaction (coupling) between nuclei (A and X) with a non-zero spin - the direct interaction (dipole-dipole coupling: D) and the indirect or scalar coupling (spin-spin splitting: J). The direct interaction is about 1000 times as large as the scalar coupling (e.g. at 2 Ã distance H-H dipolar coupling is ca 30,000 Hz). These direct couplings make the observation of high-resolution NMR spectra in solids and very viscous liquids difficult, and make NMR spectra in liquid crystals (where molecules are partially oriented, and the dipolar coupling is only partially averaged) very complex.

The scalar coupling J is a through-bond interaction, in which the spin of one nucleus perturbs (polarizes) the spins of the intervening electrons, and the energy levels of neighboring magnetic nuclei are in turn perturbed by the polarized electrons. This leads to a lowering of the energy of the neighboring nucleus when the perturbing nucleus has one spin, and a raising of the energy when it has the other spin. The J coupling (always reported in Hz) is field-independent

Coupling constants can be either positive or negative, defined as follows: coupling constants are positive if the energy of A is lower when X has the opposite spin as A, and negative if the energy of A is lower when X has the same spin as A.

PROJECT OBJECTIVE AND METHODOLOGY

The objective of our project is to predict the molecular property i.e. interaction between atoms in a molecule (scalar coupling constant) for every molecule in the test dataset by training a regression machine learning model over a given training data set.

Steps involved in the project include: 1. Data Preparation.

2. Correlation analysis to identify dependencies for the target variable.
3. Exploratory data analytics for dependent variables.
4. Modelling using regression techniques and evaluating accuracy of models.
5. Final prediction using the optimal model.

DATA SET DESCRIPTION

The dataset is obtained from one of the active Kaggle competitions " Predicting Molecular Properties" hosted by CHAMPS (CHemistry And Mathematics in Phase Space) retrieved from https://www.kaggle.com/c/champs-scalar-coupling/data (https://www.kaggle.com/c/champs-scalar-coupling/data) .

The following spreadsheets were used in our model and are as follows:

1. train.csv: this is the training data file with close to 4 million rows comprising of the molecule name, atom indices that form the pair, joint type of atoms in the molecule and their respective scalar coupling constants.
2. test.csv: this is the test file similar to that of the training file with different molecule names for which the scalar coupling constant is to be predicted.
3. structures.csv: this is the file that gives positional data of different atoms in a particular molecule by its x,y and z coordinates.
4. mulliken_charges.csv: this is the file that gives the respective mulliken charge of each atom in a molecule.
5. potential_energy.csv: this is the file that gives the potential energy of each molecule.
6. dipole_moments.csv: this is the file that gives X,Y and Z components of the dipole moments for each molecule.

IMPORTING LIBRARIES AND DATA FILES

```
library('dplyr')
```

```
## Warning: package 'dplyr' was built under R version 3.5.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library('xgboost')
```

```
## Warning: package 'xgboost' was built under R version 3.5.3
```

```
##
## Attaching package: 'xgboost'
```

```
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
library('tidyverse')
```

```
## Warning: package 'tidyverse' was built under R version 3.5.3
```

```
## -- Attaching packages ---------------------------------------------------------------
------------------------------------ tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.1     v readr   1.3.1
## v tibble  2.0.1     v purrr   0.2.5
## v tidyr   0.8.3     v stringr 1.3.1
## v ggplot2 3.1.1     v forcats 0.3.0
```

```
## Warning: package 'ggplot2' was built under R version 3.5.3
```

```
## Warning: package 'tidyr' was built under R version 3.5.3
```

```
## -- Conflicts ------------------------------------------------------------------------
-------------------------------- tidyverse_conflicts() --
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x xgboost::slice() masks dplyr::slice()
```

```
library('Metrics')
```

```
## Warning: package 'Metrics' was built under R version 3.5.3
```

```
library('ggplot2')
library('gridExtra')
```

```
## Warning: package 'gridExtra' was built under R version 3.5.3
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library('grid')
library('corrplot')
```

```
## Warning: package 'corrplot' was built under R version 3.5.3
```

```
## corrplot 0.84 loaded
```

```
train <- read.csv("C:/Users/anish/Downloads/Predictive Analytics/Final project/champs-scalar-cou
pling/train.csv")
test <- read.csv("C:/Users/anish/Downloads/Predictive Analytics/Final project/champs-scalar-coup
ling/test.csv" )
dipole <- read.csv("C:/Users/anish/Downloads/Predictive Analytics/Final project/champs-scalar-co
upling/dipole_moments.csv")
structure <- read.csv("C:/Users/anish/Downloads/Predictive Analytics/Final project/champs-scalar
-coupling/structures.csv")
potential <- read.csv("C:/Users/anish/Downloads/Predictive Analytics/Final project/champs-scalar
-coupling/potential_energy.csv")
muliken <- read.csv("C:/Users/anish/Downloads/Predictive Analytics/Final project/champs-scalar-c
oupling/mulliken_charges.csv")
```

TRAINING DATA FRAME PROPERTIES

```
summary(train)
```

```
##        id                  molecule_name      atom_index_0
##  Min.   :      0   dsgdb9nsd_042139:    135   Min.   : 0.00
##  1st Qu.:1164536   dsgdb9nsd_096580:    133   1st Qu.:11.00
##  Median :2329073   dsgdb9nsd_123246:    133   Median :13.00
##  Mean   :2329073   dsgdb9nsd_116493:    132   Mean   :13.36
##  3rd Qu.:3493610   dsgdb9nsd_121391:    132   3rd Qu.:16.00
##  Max.   :4658146   dsgdb9nsd_122665:    132   Max.   :28.00
##                    (Other)         :4657350
##   atom_index_1         type         scalar_coupling_constant
##  Min.   : 0.000   3JHC   :1510379   Min.   :-36.219
##  1st Qu.: 2.000   2JHC   :1140674   1st Qu.: -0.255
##  Median : 5.000   1JHC   : 709416   Median :  2.281
##  Mean   : 5.884   3JHH   : 590611   Mean   : 15.922
##  3rd Qu.: 8.000   2JHH   : 378036   3rd Qu.:  7.391
##  Max.   :28.000   3JHN   : 166415   Max.   :204.880
##                   (Other): 162616
```

```
str(train)
```

```
## 'data.frame':    4658147 obs. of  6 variables:
##  $ id                      : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ molecule_name           : Factor w/ 85003 levels "dsgdb9nsd_000001",..: 1 1 1 1 1 1 1 1 1
1 ...
##  $ atom_index_0            : int  1 1 1 1 2 2 2 3 3 4 ...
##  $ atom_index_1            : int  0 2 3 4 0 3 4 0 4 0 ...
##  $ type                    : Factor w/ 8 levels "1JHC","1JHN",..: 1 4 4 4 1 4 4 1 4 1 ...
##  $ scalar_coupling_constant: num  84.8 -11.3 -11.3 -11.3 84.8 ...
```

DATA PREPARATION

Theoretically, the scalar coupling constant depends upon the distance between atoms in the molecule. But given dataset has only coordinates of atoms and no distance values. Hence distances are first calculated using the given x,y and z coordinates by formula

```
sc <- train$scalar_coupling_constant
train$scalar_coupling_constant <- NULL

full <- rbind(train,test) %>%
  left_join(structure, by = c("molecule_name","atom_index_0" = "atom_index")) %>%
  left_join(structure, by = c("molecule_name","atom_index_1" = "atom_index")) %>%
  mutate(
    x_dist = x.x - x.y,
    y_dist = y.x - y.y,
    z_dist = z.x - z.y,
    dist = sqrt(x_dist^2 + y_dist^2 + z_dist^2))
```

```
## Warning: Column `molecule_name` joining factors with different levels,
## coercing to character vector
```

```
## Warning: Column `molecule_name` joining character vector and factor,
## coercing into character vector
```

```
names(full) <- c("id", "molecule_name","atom_index_0", "atom_index_1", "joint_type","atom_0","x_
0", "y_0", "z_0", "atom_1",
                 "x_1","y_1","z_1", "x_dist","y_dist", "z_dist","distance")

train_f <- full[1:nrow(train),c("molecule_name","atom_index_0","atom_index_1","joint_type","dist
ance")]
train_f$scalar_coupling_constant <- as.vector(sc)
```

EDA 1 : FILES

```
summary(train_f)
```

```
##  molecule_name       atom_index_0    atom_index_1      joint_type
##  Length:4658147     Min.   : 0.00   Min.   : 0.000   3JHC   :1510379
##  Class :character   1st Qu.:11.00   1st Qu.: 2.000   2JHC   :1140674
##  Mode  :character   Median :13.00   Median : 5.000   1JHC   : 709416
##                     Mean   :13.36   Mean   : 5.884   3JHH   : 590611
##                     3rd Qu.:16.00   3rd Qu.: 8.000   2JHH   : 378036
##                     Max.   :28.00   Max.   :28.000   3JHN   : 166415
##                                                      (Other): 162616
##     distance     scalar_coupling_constant
##  Min.   :1.002   Min.   :-36.219
##  1st Qu.:1.949   1st Qu.: -0.255
##  Median :2.313   Median :  2.281
##  Mean   :2.361   Mean   : 15.922
##  3rd Qu.:2.946   3rd Qu.:  7.391
##  Max.   :3.924   Max.   :204.880
##
```

```
summary(test)
```

```
##        id                  molecule_name      atom_index_0
##  Min.   :4658147   dsgdb9nsd_041946:    131   Min.   : 1.00
##  1st Qu.:5284532   dsgdb9nsd_060268:    131   1st Qu.:11.00
##  Median :5910918   dsgdb9nsd_060445:    131   Median :13.00
##  Mean   :5910918   dsgdb9nsd_096326:    131   Mean   :13.35
##  3rd Qu.:6537303   dsgdb9nsd_042019:    130   3rd Qu.:16.00
##  Max.   :7163688   dsgdb9nsd_059779:    130   Max.   :28.00
##                    (Other)         :2504758
##   atom_index_1         type
##  Min.   : 0.000   3JHC   :811999
##  1st Qu.: 2.000   2JHC   :613138
##  Median : 5.000   1JHC   :380609
##  Mean   : 5.878   3JHH   :317435
##  3rd Qu.: 8.000   2JHH   :203126
##  Max.   :28.000   3JHN   : 90616
##                   (Other): 88619
```

```
summary(structure)
```

```
##          molecule_name       atom_index     atom
##  dsgdb9nsd_057518:     29   Min.   : 0.000   C: 831726
##  dsgdb9nsd_058099:     29   1st Qu.: 4.000   F:   2996
##  dsgdb9nsd_058183:     29   Median : 9.000   H:1208387
##  dsgdb9nsd_059978:     29   Mean   : 8.757   N: 132361
##  dsgdb9nsd_060337:     29   3rd Qu.:13.000   O: 183187
##  dsgdb9nsd_060445:     29   Max.   :28.000
##  (Other)         :2358483
##        x                 y                z
##  Min.   :-9.23489   Min.   :-9.9339   Min.   :-9.13476
##  1st Qu.:-0.87461   1st Qu.:-1.8262   1st Qu.:-0.84249
##  Median : 0.05184   Median :-0.4036   Median : 0.01093
##  Mean   : 0.09489   Mean   :-0.3337   Mean   : 0.06241
##  3rd Qu.: 1.11610   3rd Qu.: 1.3737   3rd Qu.: 0.93944
##  Max.   : 9.38224   Max.   :10.1820   Max.   : 7.89473
##
```

EDA 2 SCALAR COUPLING CONSTANT

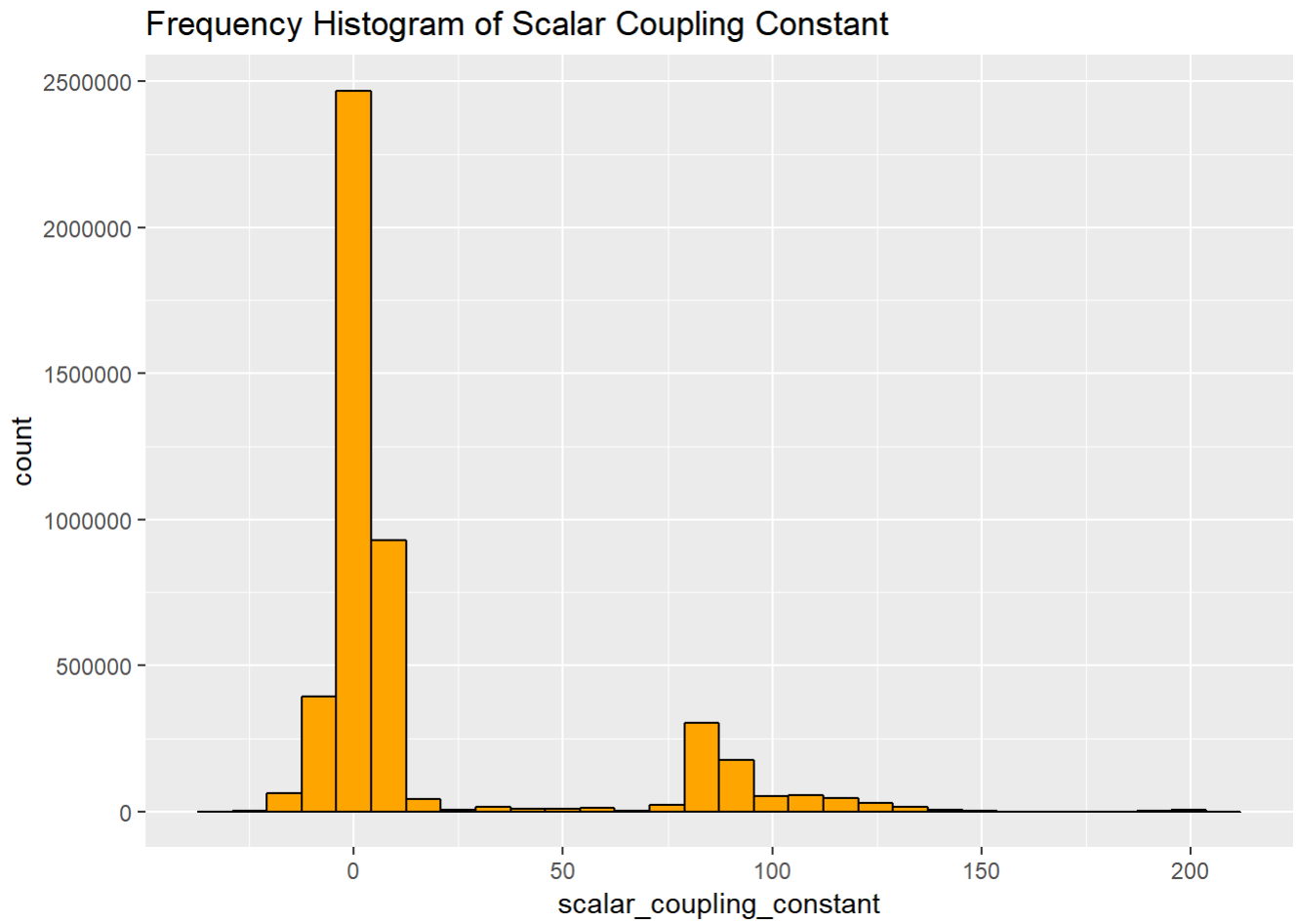
The Frequency distribution histogram of scalar coupling constant indicates that the distribution of coupling constant in training data set is bimodal which also proves the low accuracy of the model in the beginning.

EDA 3 ORIENTATION DISTRIBUTION

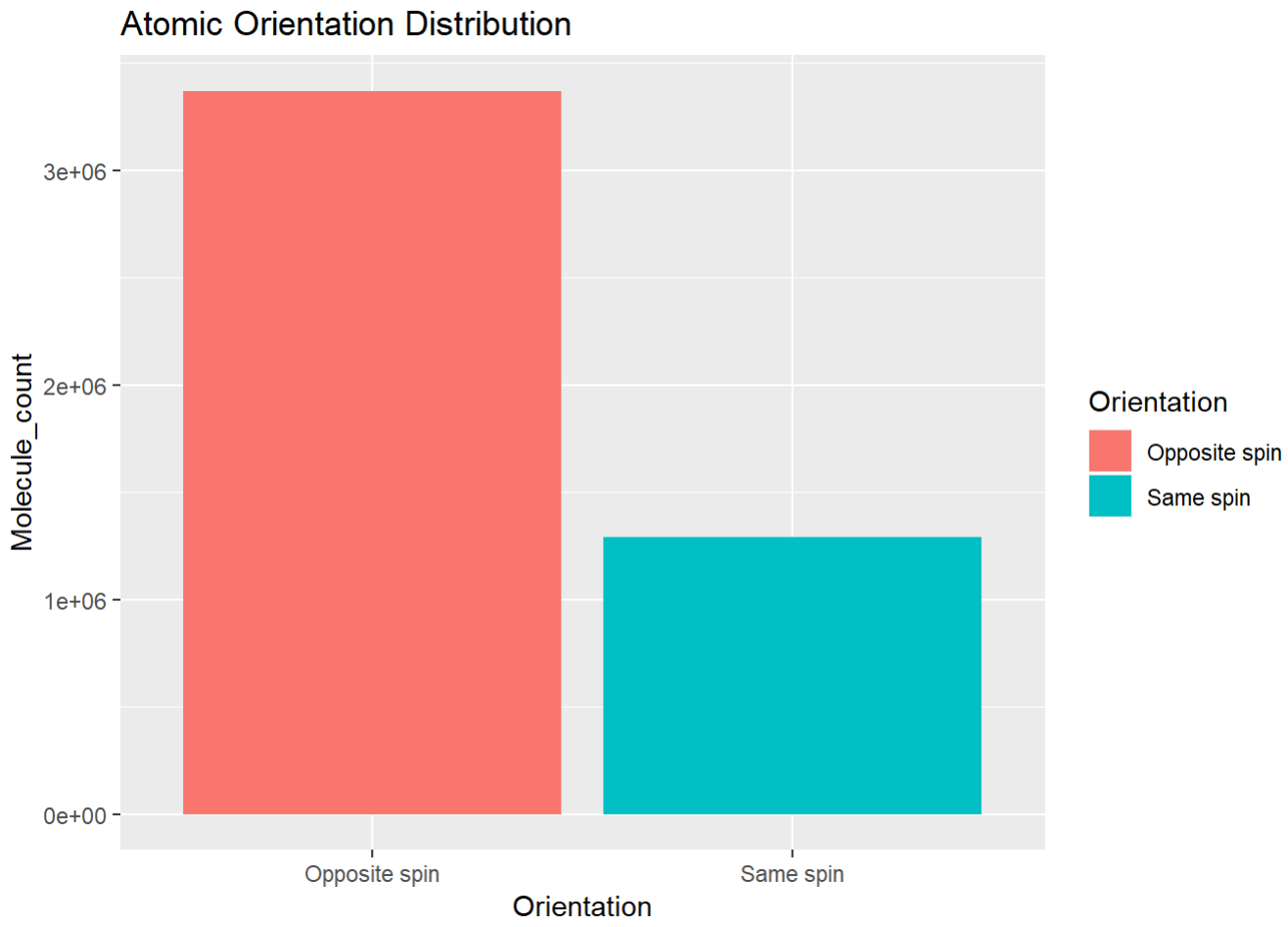
The distribution of orientation of atomic spin in a molecule has a huge difference between the same spin and opposite spin molecules and this bias can affect the accuracy of our model. Ideally distribution should be equal in the training dataset to get a model with higher accuracy.

```
#Histogram
ggplot(train_f, aes(x = scalar_coupling_constant)) +
  geom_histogram(color = "black", fill = "orange") + ggtitle("Frequency Histogram of Scalar Coup
ling Constant")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
#Bar Plot
s1 <- nrow(subset(train_f,train_f$scalar_coupling_constant<0))
s2 <- nrow(subset(train_f,train_f$scalar_coupling_constant>0))
df <- data.frame(Orientation=c("Opposite spin","Same spin"),
                 Molecule_count=c(s2,s1))
ggplot(data=df, aes(x=Orientation, y=Molecule_count,fill=Orientation)) +
  geom_bar(stat="identity") + ggtitle("Atomic Orientation Distribution")
```



EDA 4 : CORRELATION PLOT

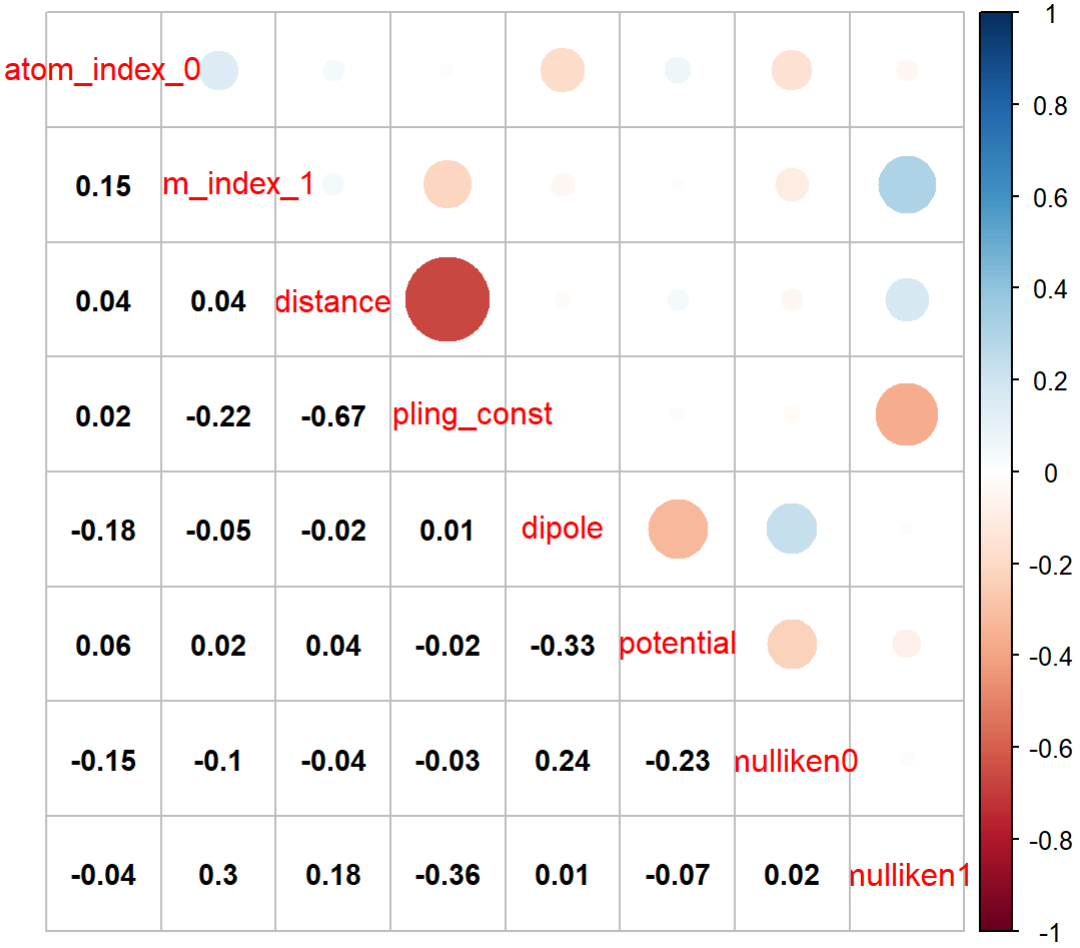
By theory and as per obtained correlation plot we observe that coupling constants depend upon distance between atoms in a molecule and hence distance becomes the most significant dependent variable for the target variable in our model.

The other variables such as mulliken charge, potential energy, dipole moments have either 0 or low correlation and hence are not considered as dependencies for the target variable in the model.

```
train1 <- full[1:nrow(train),c("molecule_name","atom_index_0","atom_index_1","joint_type",
                               "distance")]
train1$scalar_coupling_constant <- as.numeric(sc)
train2 <- train1[, c("atom_index_0", "atom_index_1","distance", "scalar_coupling_constant")]

dipole$diople_moment <- sqrt(dipole$X^2 + dipole$Y^2 + dipole$Z^2)
tr_dip <- train1 %>%
  left_join(dipole, by = c("molecule_name"))%>%
  left_join(potential, by = "molecule_name")%>%
  left_join(muliken, by = c("molecule_name", "atom_index_0" = "atom_index"))%>%
  left_join(muliken, by = c("molecule_name", "atom_index_1" = "atom_index"))
```

```
## Warning: Column `molecule_name` joining character vector and factor,
## coercing into character vector

## Warning: Column `molecule_name` joining character vector and factor,
## coercing into character vector

## Warning: Column `molecule_name` joining character vector and factor,
## coercing into character vector

## Warning: Column `molecule_name` joining character vector and factor,
## coercing into character vector
```

```
names(tr_dip) <-c ("molecule_name","atom_index_0","atom_index_1", "joint_type","distance",
                   "coupling_const", "X", "Y", "Z", "dipole",
                   "potential", "mulliken0", "mulliken1")
c <- tr_dip[, c("atom_index_0", "atom_index_1", "distance", "coupling_const",
                "dipole", "potential", "mulliken0", "mulliken1")]

d <- na.omit(c)
M <- cor(d)
corrplot.mixed(M, lower.col ="black", number.cex=0.9, upper = "circle")
```



EDA 5 : Scalar Coupling Constant Distribution by Atomic Type

The distribution of coupling constant for different joint types are either right or left skewed with minor peaks in the skew direction.

1JHC :

. This is a molecule made by a single bond between a hydrogen and carbon atom. . There are 709416 molecules of this combination in the data set which is distributed in the data set with right skew. . The molecules of this atomic type have positive coupling constants only i.e. molecules of this atomic type have atoms spinning only in opposite directions

2JHC :

. This is a molecule made by a double bond between a hydrogen and carbon atom. . There are 1140674 molecules of this combination in the data set which is symmetrically distributed. . This atomic type molecules have equal no.of positive and negative coupling constants i.e. molecules of this atomic type have atoms spinning in both opposite and parallel directions

3JHC :

. This is a molecule made by a triple bond between a hydrogen and carbon atom. . There are 1510379 molecules of this combination in the data set which is the highest and normally distributed in the data set with right skew. . The molecules of this atomic type have positive coupling constants only i.e. molecules of this atomic type have atoms spinning only in opposite directions and the difference from 1JHC is that the magnitude of coupling constant is very low in triple bond compared to single bond.

2JHH :

. This is a molecule made by a double bond between a two hydrogen atoms. . There are 378036 molecules of this combination in the data set which is symmetrically distributed with peak in negative scale. . The molecules of this atomic type have a majority of negative coupling constants i.e. a majority of the molecules of this atomic type have atoms spinning in parallel directions

3JHH :

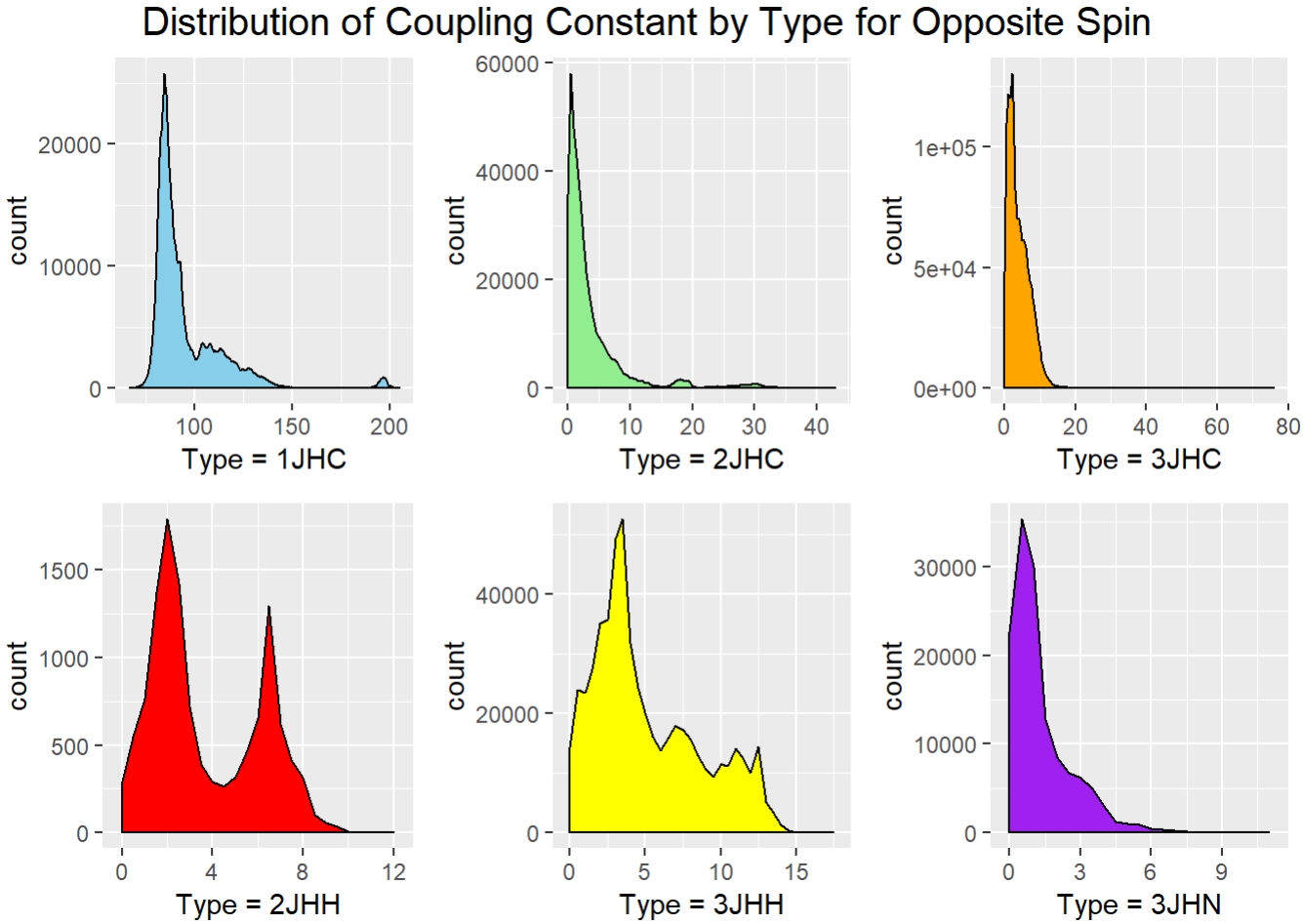
. This is a molecule made by a triple bond between a two hydrogen atoms. . There are 590611 molecules of this combination in the data set which is distributed with many peaks skewing towards the right. . The molecules of this atomic type have a majority of positive coupling constants i.e. a majority of the molecules of this atomic type have atoms spinning in opposite directions.

3JHN :

. This is a molecule made by a triple bond between a two hydrogen atoms. . There are 166415 molecules of this combination in the data set which is distributed with many peaks skewing towards the right. . The molecules of this atomic type have positive coupling constants only i.e. molecules of this atomic type have atoms spinning only in opposite directions and the difference from 3JHC is that the magnitude of coupling constant is very low in triple bond with nitrogen than bond with carbon.

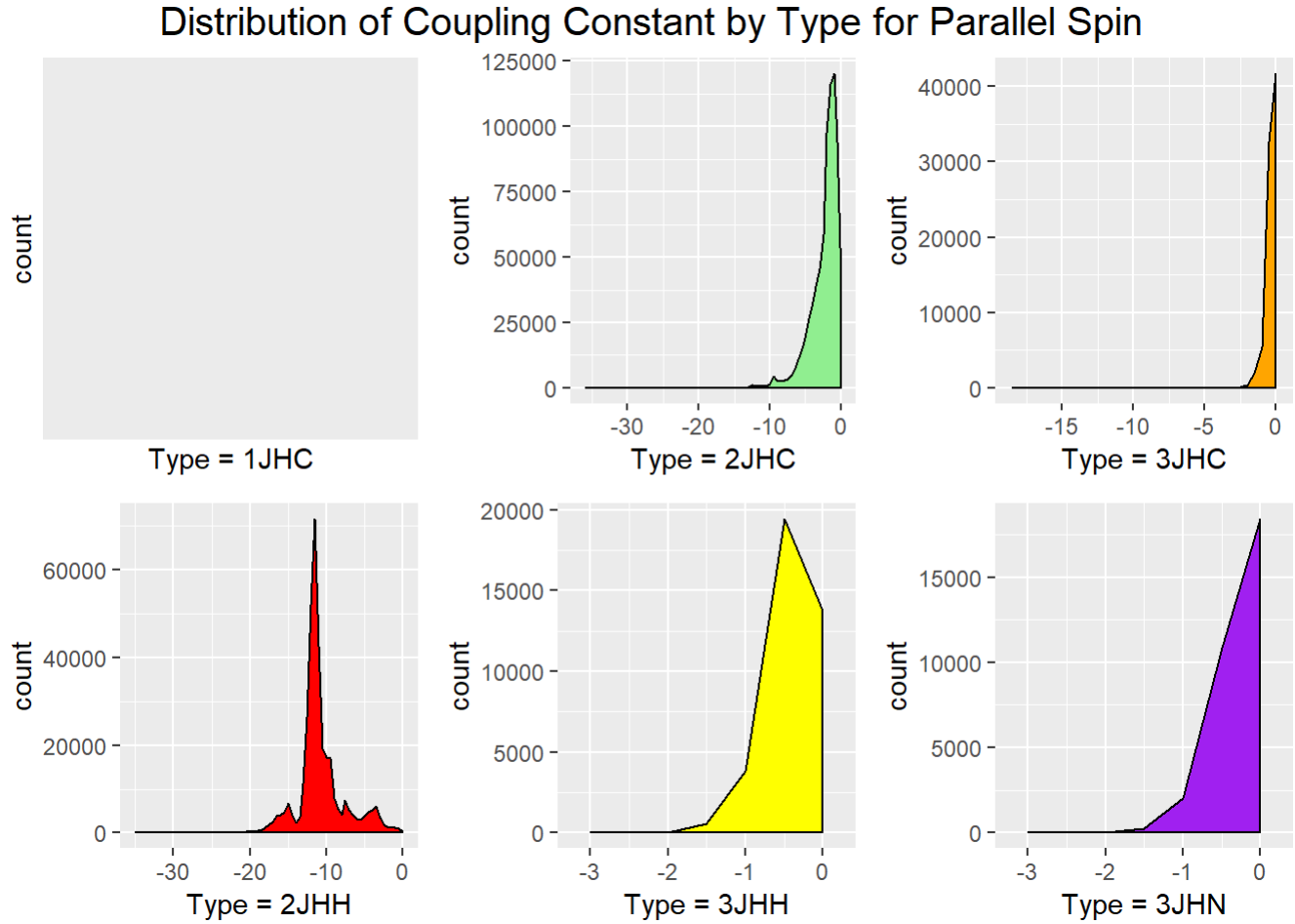
```
p1 <- ggplot(filter(train_f, scalar_coupling_constant > 0,joint_type == "1JHC"), aes(x = scalar_
coupling_constant)) +
      geom_area(stat = "bin",
            binwidth = 0.5,
            colour = "black",
            fill = "skyblue",
            linetype = "solid") +
      labs(x="Type = 1JHC")

p2 <- ggplot(filter(train_f,scalar_coupling_constant > 0, joint_type == "2JHC"), aes(x = scalar_
coupling_constant)) +
  geom_area(stat = "bin",
            binwidth = 0.5,
            colour = "black",
            fill = "lightgreen",
            linetype = "solid") +
  labs(x="Type = 2JHC")


p3 <- ggplot(filter(train_f,scalar_coupling_constant > 0, joint_type == "3JHC"), aes(x = scalar_
coupling_constant)) +
  geom_area(stat = "bin",
            binwidth = 0.5,
            colour = "black",
            fill = "orange",
            linetype = "solid") +
  labs(x="Type = 3JHC")

p4 <- ggplot(filter(train_f, scalar_coupling_constant > 0,joint_type == "2JHH"), aes(x = scalar_
coupling_constant)) +
  geom_area(stat = "bin",
            binwidth = 0.5,
            colour = "black",
            fill = "red",
            linetype = "solid") +
  labs(x="Type = 2JHH")


p5 <- ggplot(filter(train_f,scalar_coupling_constant > 0, joint_type == "3JHH"), aes(x = scalar_
coupling_constant)) +
  geom_area(stat = "bin",
            binwidth = 0.5,
            colour = "black",
            fill = "yellow",
            linetype = "solid") +
  labs(x="Type = 3JHH")


p6 <- ggplot(filter(train_f,scalar_coupling_constant > 0, joint_type == "3JHN"), aes(x = scalar_
coupling_constant)) +
  geom_area(stat = "bin",
            binwidth = 0.5,
            colour = "black",
```

```
            fill = "purple",
            linetype = "solid") +
  labs(x="Type = 3JHN")

grid.arrange(p1,p2,p3,p4,p5,p6, ncol=3,
             top = textGrob("Distribution of Coupling Constant by Type for Opposite Spin",gp=gpa
r(fontsize=15)))
```



```
p7 <- ggplot(filter(train_f, scalar_coupling_constant < 0,joint_type == "1JHC"), aes(x = scalar_
coupling_constant)) +
      geom_area(stat = "bin",
            binwidth = 0.5,
            colour = "black",
            fill = "skyblue",
            linetype = "solid") +
      labs(x="Type = 1JHC")

p8 <- ggplot(filter(train_f,scalar_coupling_constant < 0, joint_type == "2JHC"), aes(x = scalar_
coupling_constant)) +
  geom_area(stat = "bin",
            binwidth = 0.5,
            colour = "black",
            fill = "lightgreen",
            linetype = "solid") +
  labs(x="Type = 2JHC")


p9 <- ggplot(filter(train_f,scalar_coupling_constant < 0, joint_type == "3JHC"), aes(x = scalar_
coupling_constant)) +
  geom_area(stat = "bin",
            binwidth = 0.5,
            colour = "black",
            fill = "orange",
            linetype = "solid") +
  labs(x="Type = 3JHC")

p10 <- ggplot(filter(train_f, scalar_coupling_constant < 0,joint_type == "2JHH"), aes(x = scalar
_coupling_constant)) +
  geom_area(stat = "bin",
            binwidth = 0.5,
            colour = "black",
            fill = "red",
            linetype = "solid") +
  labs(x="Type = 2JHH")


p11 <- ggplot(filter(train_f,scalar_coupling_constant < 0, joint_type == "3JHH"), aes(x = scalar
_coupling_constant)) +
  geom_area(stat = "bin",
            binwidth = 0.5,
            colour = "black",
            fill = "yellow",
            linetype = "solid") +
  labs(x="Type = 3JHH")


p12 <- ggplot(filter(train_f,scalar_coupling_constant < 0, joint_type == "3JHN"), aes(x = scalar
_coupling_constant)) +
  geom_area(stat = "bin",
            binwidth = 0.5,
            colour = "black",
```

```
            fill = "purple",
            linetype = "solid") +
  labs(x="Type = 3JHN")

grid.arrange(p7,p8,p9,p10,p11,p12, ncol=3,
             top = textGrob("Distribution of Coupling Constant by Type for Parallel Spin",gp=gpa
r(fontsize=15)))
```



MODELLING THE DATA:

From the above correlation plot and EDA of coupling constant for different types, we infer that for the target variable i.e. scalar coupling constant, the significant dependent variables are atomic distance and joint type of the atoms in the molecule. Based on these 2 dependencies we implement two models in our training dataset, evaluate the accuracy parameters of the models with the predicted values using the training dataset and use the models to predict scalar coupling constant in the test dataframe.

Model 1: Linear Regression

Linear Regression is the methood used to determine the values of dependent variable(scalar coupling constant) using the independent variables(distance and joint type). It indicates that the reponse variable is linearly dependent on the independent variable in the form of an equation: y = mx + c where, c is the intercept value and m

is the cofficient of the independent variable x and y being the dependent variable. the dependency of x and y can be positive/negative, that is, if an incement of x leads to increase inthe value of y hten they are positively correlated and vice-versa.

From EDA of the response variable, scalar coupling constant, there are sufficient evidences for its dependence on atomic distance which is in turn affected by the type of joint between the atoms. Using the 'distance' parameter calculated in data preparation and the 'joint type' variable, generated a linear regression model to determine the dependency of scalar coupling constant on these variables. Using lm() function is R, we get a p-value of 2e-16, verifying that the independent factors strongly affect the J - interaction between the atoms. The obtained estimated coefficients are used to get the model equation and compute the values for new instances of the test set. We get an adjusted R-square value of 94%, which further convinces us about the reliability of the model. The mean absolute error

```
reg <- lm(scalar_coupling_constant~ distance + joint_type , data = train_f)
summary(reg)
```

```
##
## Call:
## lm(formula = scalar_coupling_constant ~ distance + joint_type,
##     data = train_f)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -36.197  -2.655  -0.785   1.653 110.003
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      91.33817    0.02103  4343.9   <2e-16 ***
## distance          3.32874    0.01720   193.5   <2e-16 ***
## joint_type1JHN  -47.22985    0.03928 -1202.3   <2e-16 ***
## joint_type2JHC  -98.90006    0.02237 -4421.5   <2e-16 ***
## joint_type2JHH -107.53295    0.01982 -5424.5   <2e-16 ***
## joint_type2JHN  -95.32335    0.03064 -3111.2   <2e-16 ***
## joint_type3JHC  -97.89906    0.03602 -2717.9   <2e-16 ***
## joint_type3JHH  -95.56181    0.03101 -3081.9   <2e-16 ***
## joint_type3JHN -100.50136    0.04001 -2511.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.936 on 4658138 degrees of freedom
## Multiple R-squared:  0.9484, Adjusted R-squared:  0.9484
## F-statistic: 1.071e+07 on 8 and 4658138 DF,  p-value: < 2.2e-16
```

```
train_f$pred <- predict(reg, newdata = train_f)
test1 <- full[(nrow(train)+1):nrow(full),]
test1$predlm <- predict(reg, newdata = test1)
prediction_lm <- test1$predlm
```

Model 2: Gradient Boosting

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of and ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.This algorithm has base model is based on creation of decision trees. IN this method initially a tree model is produced and predicitons are made using it. The error calculated using those predicted values is used to adjust the predictions to train the next models using lazy learning technique. Lazy learning is an ensemble of weakly trained (adjusted) models which eventually lead to a strong model genration which produces least possible errors while predicting the response variable for the test set.

The objective parameter used in gradient boosting is linear regression with evaluation parameter being mae (mean absolute error). The number of iterations, that is, nrounds is 100 and the max depth of the tree is kept 0.2. the learning rate, which defines the proportion of last model output to be included in the next model, is kept at 0.5 after many test values such as, 0.01, 0.1, 0.3, and finally 0.5. In this model, a regression tree is being trained repeated using lazy learning technique.

```
train1 <- xgb.DMatrix(data = as.matrix(train_f[, c("distance")]), label = train_f$scalar_couplin
g_constant)
test2 <- xgb.DMatrix(data = as.matrix(test1[, c("distance")]))
xgb_params <- list("objective" = "reg:linear",
                   "eval_metric" = "mae")

model <- xgb.train(params = xgb_params,
                   data = train1,
                   eta = 0.5,
                   nrounds = 100,
                   max_depth = 2,
                   subsample = 0.9,
                   colsample_bytree = 1)

train_pred_xgb <- predict(model,  newdata = train1)
prediction_xgb <- predict(model, newdata = test2)
```

Error Calculations

The mean absolute error observed in the linear regression model is 4.134 The mean absolute error obtained in gradient boosting model is 4.221 Thus, we observed that the mean absolute error is increasing in the case of gradient boosting

```
mae <- function(error)
{
  mean(abs(error))
}
mae(train_f$pred-train_f$scalar_coupling_constant)
```

```
## [1] 4.134075
```

```
mae(train_pred_xgb-train_f$scalar_coupling_constant)
```

```
## [1] 4.231448
```

Final Predictions

After Accuracy evaluation we have used both the models to predict scalar coupling constant in the test datset and combined both model predictions into a single dataframe.

```
final <- test1$molecule_name
final <- cbind(final,prediction_lm)
final <- cbind(final,prediction_xgb)
head(final)
```

```
##      final              prediction_lm         prediction_xgb
## [1,] "dsgdb9nsd_000004" "-0.0350117184739105" "3.29490518569946"
## [2,] "dsgdb9nsd_000004" "94.8736262229335"    "88.4286422729492"
## [3,] "dsgdb9nsd_000004" "6.83868834158255"    "4.13365316390991"
## [4,] "dsgdb9nsd_000004" "94.8736262229335"    "88.4286422729492"
## [5,] "dsgdb9nsd_000004" "-0.0350117184739105" "3.29490518569946"
## [6,] "dsgdb9nsd_000015" "95.0075375573793"    "61.1222877502441"
```

Conclusions

Altering the ‘xgb’ parameters, like objective, nrounds, eta, max depth, and adding new parameters like min_child_weight, base_score, best_score, feature_names would help to improve accuracy by reducing the mean absolute error values.

Adding the effect of other variables such as mulliken charges, dipole moments, and potential energy will also, improve the accuracy but we may run into the possibility of creating an overfitted model. Moreover, adding interaction variables with the given variables to train the model will also reduce the mean absolute error values of the entire model. This is beacuse the variables are not completely independent of each other.

Since, the error value in the performed xgb inceases, making linear model to be more accurate with the given parameters.

REFERENCE

[1]Hans J. Reich (2017), Spin-Spin Splitting: J-Coupling, Retrieved from https://www.chem.wisc.edu/areas/reich/nmr/Notes-05-HMR-v26-part2.pdf (https://www.chem.wisc.edu/areas/reich/nmr/Notes-05-HMR-v26-part2.pdf)