Similarity

CS 4375 - Intro to Machine Learning

Dr. Karen Mazidi

Author:

- Leo Nguyen - ldn190002
- Simon Kim - sxk190106
- Abed Ahmed - asa190005
- Dylan Kapustka - dlk190000

Physicochemical Properties of Protein Tertiary Structure Data Set

Citation:

ABV - Indian Institute of Information Technology & Management, Gwalior, MP, India. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml (http://archive.ics.uci.edu/ml)]. Irvine, CA: University of California, School of Information and Computer Science.

Link for Data Set
https://archive.ics.uci.edu/ml/datasets/Physicochemical+Properties+of+Protein+Tertiary+Structure (https://archive.ics.uci.edu/ml/datasets/Physicochemical+Properties+of+Protein+Tertiary+Structure)

Attribute Information:

- RMSD-Size of the residue.
- F1 - Total surface area.
- F2 - Non polar exposed area.
- F3 - Fractional area of exposed non polar residue.
- F4 - Fractional area of exposed non polar part of residue.
- F5 - Molecular mass weighted exposed area.
- F6 - Average deviation from standard exposed area of residue.
- F7 - Euclidean distance.
- F8 - Secondary structure penalty.
- F9 - Spacial Distribution constraints (N,K Value).

Load the data

```
df <- read.csv("data/CASP.csv", header=TRUE)
str(df)
```

```
## 'data.frame':    45730 obs. of  10 variables:
##  $ RMSD: num  17.28 6.02 9.28 15.85 7.96 ...
##  $ F1  : num  13558 6192 7726 8425 7461 ...
##  $ F2  : num  4305 1623 1726 2368 1737 ...
##  $ F3  : num  0.318 0.262 0.223 0.281 0.233 ...
##  $ F4  : num  162.2 53.4 67.3 67.8 52.4 ...
##  $ F5  : num  1872791 803447 1075648 1210472 1021020 ...
##  $ F6  : num  215.4 87.2 81.8 109.4 94.5 ...
##  $ F7  : num  4288 3329 2981 3248 2814 ...
##  $ F8  : int  102 39 29 70 41 15 70 74 39 26 ...
##  $ F9  : num  27 38.5 38.8 39.1 39.9 ...
```

Data Cleaning

Remove the column RMSD

```
df = subset(df, select = -c(RMSD))
str(df)
```

```
## 'data.frame':    45730 obs. of  9 variables:
##  $ F1: num  13558 6192 7726 8425 7461 ...
##  $ F2: num  4305 1623 1726 2368 1737 ...
##  $ F3: num  0.318 0.262 0.223 0.281 0.233 ...
##  $ F4: num  162.2 53.4 67.3 67.8 52.4 ...
##  $ F5: num  1872791 803447 1075648 1210472 1021020 ...
##  $ F6: num  215.4 87.2 81.8 109.4 94.5 ...
##  $ F7: num  4288 3329 2981 3248 2814 ...
##  $ F8: int  102 39 29 70 41 15 70 74 39 26 ...
##  $ F9: num  27 38.5 38.8 39.1 39.9 ...
```

Divide into 80/20 train/test

```
set.seed(1234)
i <- sample(1:nrow(df), 0.8*nrow(df), replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Data Exploration

1. Look at the first 10 rows in training data

```
head(train, n=10)
```

```
##             F1      F2      F3       F4        F5       F6      F7  F8      F9
## 40784  7085.06 1713.77 0.24188  71.0054  964112.9  94.5983 3096.25  44 36.6171
## 40854 10964.20 3078.91 0.28081 119.8650 1608625.6 141.0520 3745.58 139 36.0505
## 41964  6926.30 2033.94 0.29365  69.2471  956051.2  99.5273 3180.47  37 34.9845
## 15241 15273.60 4826.20 0.31598 160.1420 2116707.0 233.5890 5339.67 200 26.5049
## 33702  6999.15 1648.37 0.23551  70.6835  949550.4  93.3331 3183.40  51 37.6965
## 35716  8493.93 3658.40 0.43070  80.7255 1202937.5 138.3440 3407.53  41 37.1324
## 17487  4984.98 1395.35 0.27991  43.4339  686591.4  63.0595 2810.20  25 41.5209
## 15220  9389.75 2156.41 0.22965 114.0400 1306068.5 128.4600 4221.14  51 34.1424
## 19838 15649.50 5251.22 0.33555 218.4740 2175659.5 256.4370 5372.32 287 24.3666
## 2622   4120.18 1641.61 0.39843  38.3952  555175.6  56.8550 1401.72  52 44.2281
```

2. View the summary of entire training data set

```
summary(train)
```

```
##        F1              F2               F3               F4
##  Min.   : 2392   Min.   :  403.5   Min.   :0.09362   Min.   : 10.69
##  1st Qu.: 6939   1st Qu.: 1979.1   1st Qu.:0.25854   1st Qu.: 63.59
##  Median : 8896   Median : 2666.4   Median :0.29985   Median : 87.66
##  Mean   : 9873   Mean   : 3014.8   Mean   :0.30211   Mean   :103.59
##  3rd Qu.:12136   3rd Qu.: 3783.4   3rd Qu.:0.34260   3rd Qu.:133.87
##  Max.   :40035   Max.   :15061.6   Max.   :0.57769   Max.   :369.32
##        F5                F6               F7              F8
##  Min.   : 319490   Min.   : 31.97   Min.   :     0   Min.   :  0.00
##  1st Qu.: 954062   1st Qu.: 94.60   1st Qu.:  3165   1st Qu.: 31.00
##  Median :1236808   Median :126.20   Median :  3840   Median : 54.00
##  Mean   :1368628   Mean   :145.59   Mean   :  3982   Mean   : 69.95
##  3rd Qu.:1691223   3rd Qu.:181.36   3rd Qu.:  4645   3rd Qu.: 91.00
##  Max.   :5472011   Max.   :598.41   Max.   :105948   Max.   :350.00
##        F9
##  Min.   :15.23
##  1st Qu.:30.39
##  Median :35.29
##  Mean   :34.51
##  3rd Qu.:38.86
##  Max.   :55.07
```

3. Count number of NA value by column

```
colSums(is.na(train))
```

```
## F1 F2 F3 F4 F5 F6 F7 F8 F9
##  0  0  0  0  0  0  0  0  0
```

4. Check the correlation of all column in training data

- Based on the number we can see that F1 is highly correlation with F2, F4, F5, F6, F9

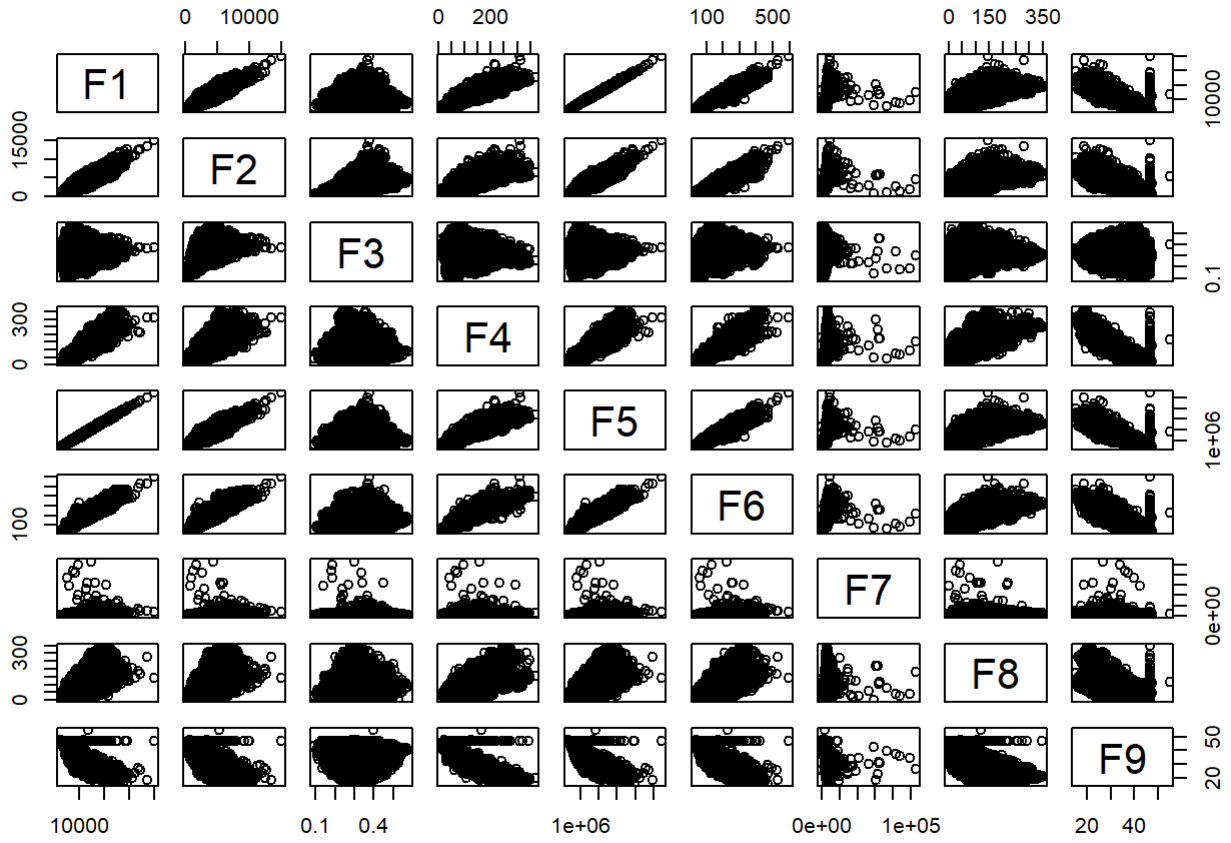
```
cor(train)
```

```
##            F1         F2          F3          F4         F5         F6
## F1  1.0000000  0.9061435  0.12558257  0.93175812  0.9981952  0.9675916
## F2  0.9061435  1.0000000  0.50305999  0.79388071  0.9024772  0.9072965
## F3  0.1255826  0.5030600  1.00000000  0.03130315  0.1219910  0.1988397
## F4  0.9317581  0.7938807  0.03130315  1.00000000  0.9264458  0.9390220
## F5  0.9981952  0.9024772  0.12199103  0.92644583  1.0000000  0.9618448
## F6  0.9675916  0.9072965  0.19883974  0.93902201  0.9618448  1.0000000
## F7  0.5664156  0.5266875  0.08042454  0.49927458  0.5659698  0.5493124
## F8  0.6515538  0.5848816  0.09619772  0.67717518  0.6431753  0.6632429
## F9 -0.9009766 -0.7898221 -0.07039026 -0.89274895 -0.9004213 -0.8850988
##             F7          F8          F9
## F1  0.56641558  0.65155381 -0.90097656
## F2  0.52668748  0.58488158 -0.78982213
## F3  0.08042454  0.09619772 -0.07039026
## F4  0.49927458  0.67717518 -0.89274895
## F5  0.56596977  0.64317528 -0.90042132
## F6  0.54931241  0.66324295 -0.88509884
## F7  1.00000000  0.36007938 -0.53716251
## F8  0.36007938  1.00000000 -0.63935345
## F9 -0.53716251 -0.63935345  1.00000000
```

5. Quick visualization on correlation

- These plots confirm the highly correlation of F1 and other attributes mentioned above

```
pairs(train)
```



6. Summary() for F1 and F5 in training data

Summary for F1 - Total surface area.

```
summary(train$F1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2392    6939    8896    9873   12136   40035
```

Summary for F5 - Molecular mass weighted exposed area.

```
summary(train$F5)
```

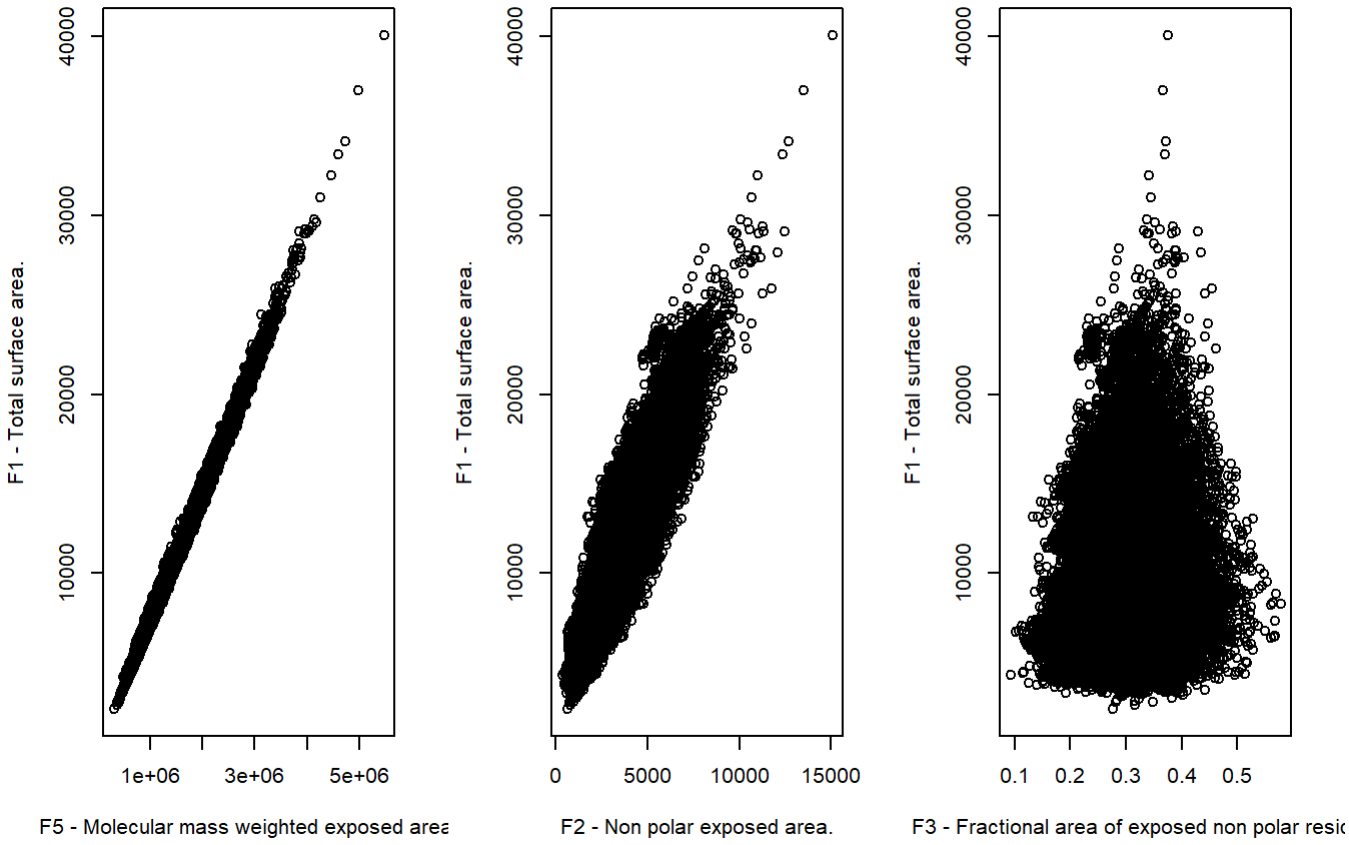
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  319490  954062 1236808 1368628 1691223 5472011
```

Data Visualization using informative graph

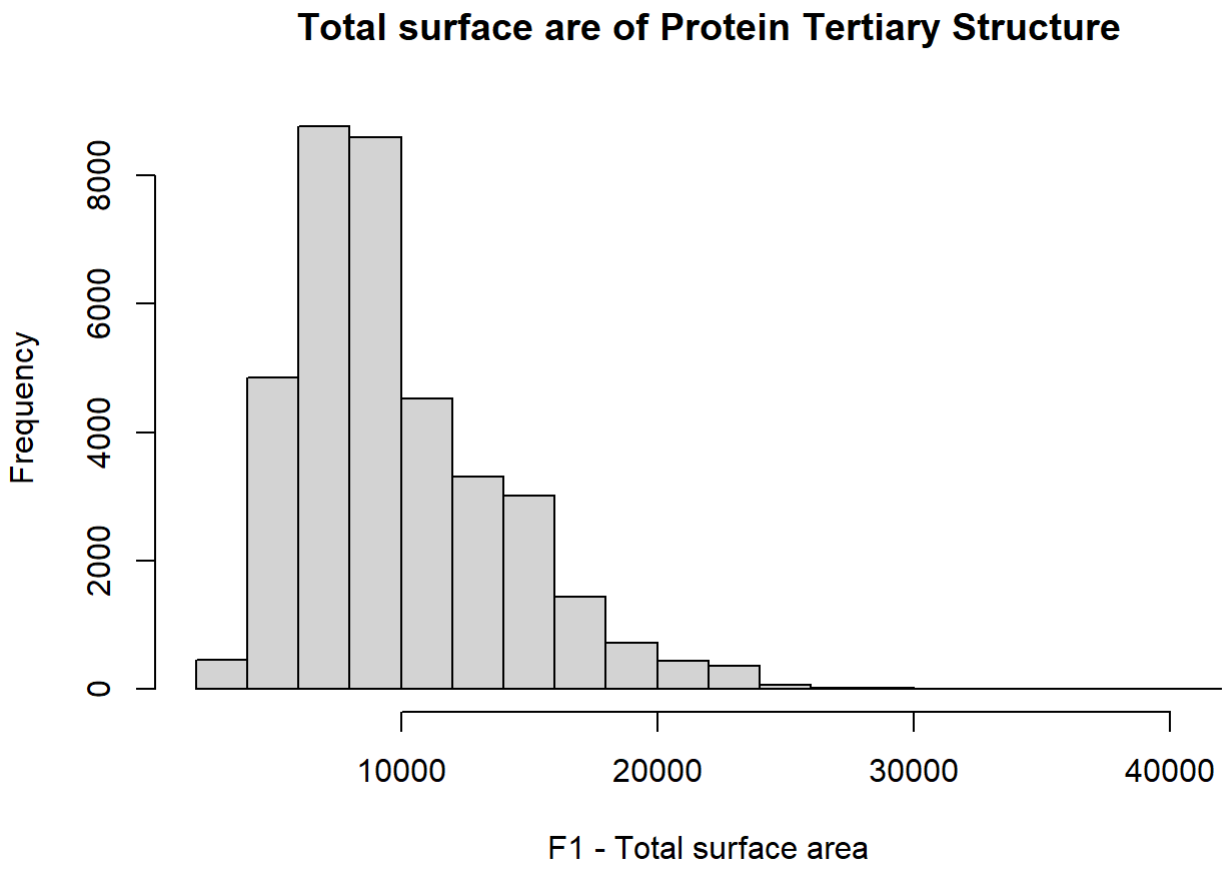
1. Plot show the relationship between F1 - Total surface area with:

- F5 - Molecular mass weighted exposed area. (Most correlation with F1)
- F2 - Non polar exposed area. (Some correlation with F1)
- F3 - Fractional area of exposed non polar residue. (Leasst correclation with F1)

```
par(mfrow=c(1,3))
plot(train$F5,train$F1 ,xlab = "F5 - Molecular mass weighted exposed area.", ylab = "F1 - Total
 surface area." )
plot(train$F2,train$F1 ,xlab = "F2 - Non polar exposed area.", ylab = "F1 - Total surface area."
)
plot(train$F3,train$F1 ,xlab = "F3 - Fractional area of exposed non polar residue.", ylab = "F1
 - Total surface area." )
```




```
hist(train$F1, xlab = "F1 - Total surface area", main = "Total surface are of Protein Tertiary S
tructure")
```



Comparing 3 models using correlation metric and mse

Linear Regression Model using all other column as predictors

Build and output the summary

```
lm <- lm(F1~., data=train)
summary(lm)
```

```
##
## Call:
## lm(formula = F1 ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1069.45  -120.84    -8.85   112.10  1447.48
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.049e+03  2.444e+01  42.926  < 2e-16 ***
## F2           3.570e-01  4.406e-03  81.031  < 2e-16 ***
## F3          -3.210e+03  4.214e+01 -76.172  < 2e-16 ***
## F4           2.610e+00  6.820e-02  38.273  < 2e-16 ***
## F5           5.717e-03  1.172e-05 487.739  < 2e-16 ***
## F6           3.473e+00  7.200e-02  48.229  < 2e-16 ***
## F7           2.179e-03  6.606e-04   3.299  0.00097 ***
## F8           5.461e-01  2.504e-02  21.808  < 2e-16 ***
## F9           2.039e+00  4.281e-01   4.764 1.91e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 196.6 on 36575 degrees of freedom
## Multiple R-squared:  0.9977, Adjusted R-squared:  0.9977
## F-statistic: 1.942e+06 on 8 and 36575 DF,  p-value: < 2.2e-16
```

Calculate correlation, mse and rmse for Linear Regression lm

```
pred_lm <- predict(lm, newdata=test)
cor_lm <- cor(pred_lm, test$F1)
mse_lm <- mean((pred_lm-test$F1)^2)
rmse_lm <- sqrt(mse_lm)
print(paste('correlation:', cor_lm))
```

```
## [1] "correlation: 0.998812915661555"
```

```
print(paste('mse:', mse_lm))
```

```
## [1] "mse: 39255.1941672328"
```

```
print(paste('rmse:', rmse_lm))
```

```
## [1] "rmse: 198.129236023442"
```

kNN Regression

Build kNN Regression Model

Load the require library for kNN Regression**

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

First we need to find the best k value

```
cor_k <- rep(0, 20)
mse_k <- rep(0, 20)
i <- 1
for (k in seq(1, 39, 2)){
  fit_k <- knnreg(train[,2:9],train[,1],k=k)
  pred_k <- predict(fit_k, test[,2:9])
  cor_k[i] <- cor(pred_k, test$F1)
  mse_k[i] <- mean((pred_k - test$F1)^2)
  print(paste("k=", k, cor_k[i], mse_k[i]))
  i <- i + 1
}
```

```
## [1] "k= 1 0.997469235502152 83709.095203127"
## [1] "k= 3 0.998008972762981 65863.7732417383"
## [1] "k= 5 0.998078347521015 63589.2584474594"
## [1] "k= 7 0.998035057310232 65051.3429673255"
## [1] "k= 9 0.998004585736964 66106.9215837029"
## [1] "k= 11 0.997976531803893 67064.0172264996"
## [1] "k= 13 0.997928962225012 68666.5801431176"
## [1] "k= 15 0.997892037244687 69907.6250138419"
## [1] "k= 17 0.997841958540308 71611.187673205"
## [1] "k= 19 0.997797333893946 73126.6553611172"
## [1] "k= 21 0.997762544348237 74299.5045620787"
## [1] "k= 23 0.997726835965071 75490.1793430862"
## [1] "k= 25 0.997685994516625 76850.6929169408"
## [1] "k= 27 0.997659559745927 77728.1525894446"
## [1] "k= 29 0.99762719870553 78810.1660777073"
## [1] "k= 31 0.997601630406737 79687.7520173741"
## [1] "k= 33 0.997569269264915 80787.2230712938"
## [1] "k= 35 0.99754127429365 81721.8883287005"
## [1] "k= 37 0.997511771452442 82719.1071373071"
## [1] "k= 39 0.997472435568705 84033.6326299745"
```

Some visualization on the output

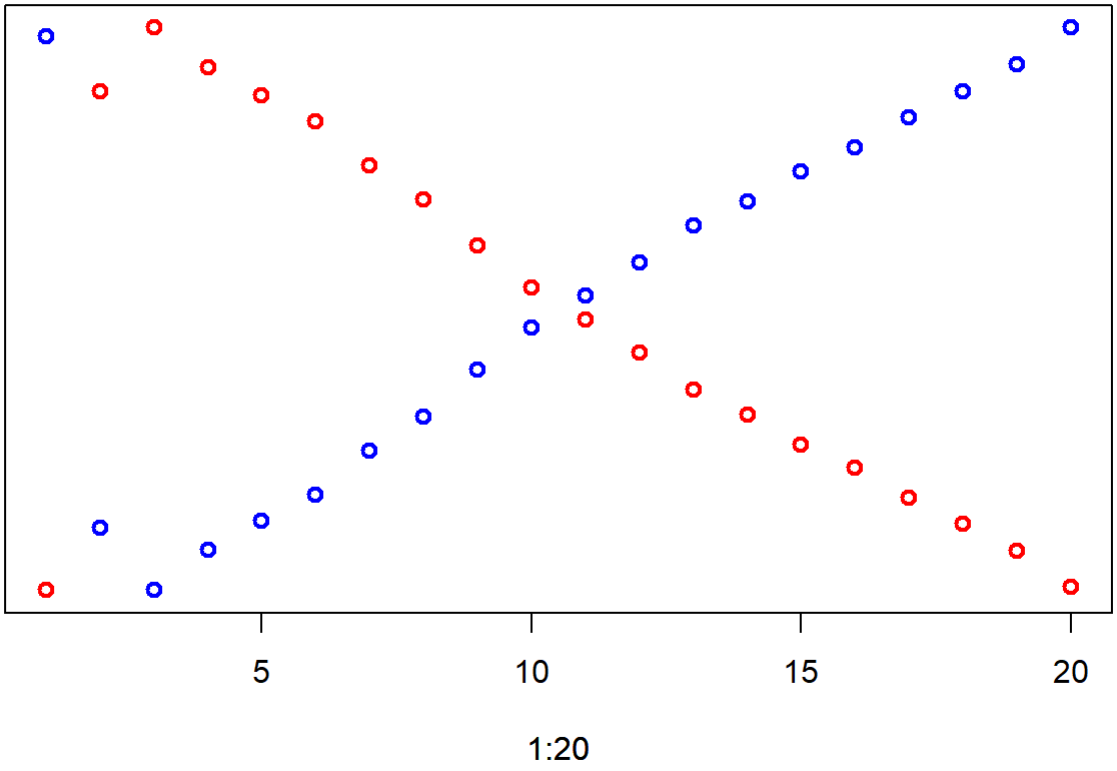
```
plot(1:20, cor_k, lwd=2, col='red', ylab="", yaxt='n')
par(new=TRUE)
plot(1:20, mse_k, lwd=2, col='blue', labels=FALSE, ylab="", yaxt='n')
```

```
## Warning in plot.window(...): "labels" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "labels" is not a graphical parameter
```

```
## Warning in box(...): "labels" is not a graphical parameter
```

```
## Warning in title(...): "labels" is not a graphical parameter
```



The best k value is 3

```
which.min(mse_k)
```

```
## [1] 3
```

```
which.max(cor_k)
```

```
## [1] 3
```

We fit the model using all predicor with k=3

```
fit <- knnreg(train[,2:9],train[,1],k=3)
```

Calculate correlation, mse and rmse for kNN Regression

```
pred_knn <- predict(fit, test[,2:9])
cor_knn <- cor(pred_knn, test$F1)
mse_knn <- mean((pred_knn - test$F1)^2)
rmse_knn <- sqrt(mse_knn)
print(paste('correlation:', cor_knn))
```

```
## [1] "correlation: 0.998008972762981"
```

```
print(paste('mse:', mse_knn))
```

```
## [1] "mse: 65863.7732417383"
```

```
print(paste('rmse:', rmse_knn))
```

```
## [1] "rmse: 256.639383652896"
```

Decision Tree Regression

Build Decision Regression Model

Load require library for decision tree model

```
library(tree)
```

Perform the decision tree regression model with all predictors. As we can see the model is only actually use F5 as the predictor. It is because tree use some indication such as entropy, information gain and Gini index to select a good information variables. This confirm the information we have when we exploring the data above (Check the graph between F1 vs F2, F5, F3)

```
tree <- tree(F1~., data=train)
summary(tree)
```

```
##
## Regression tree:
## tree(formula = F1 ~ ., data = train)
## Variables actually used in tree construction:
## [1] "F5"
## Number of terminal nodes:  8
## Residual mean deviance:  524100 = 1.917e+10 / 36580
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -3152.0  -523.4   -14.2     0.0   503.6 17490.0
```

Calculate correlation, mse and rmse for Decision Tree Regression

```
pred_tree <- predict(tree, newdata=test)
cor_tree <- cor(pred_tree, test$F1)
mse_tree <- mean((pred_tree - test$F1)^2)
rmse_tree <- sqrt(mse_tree)
print(paste('correlation:', cor_tree))
```

```
## [1] "correlation: 0.981994447324154"
```

```
print(paste('mse:', mse_tree))
```

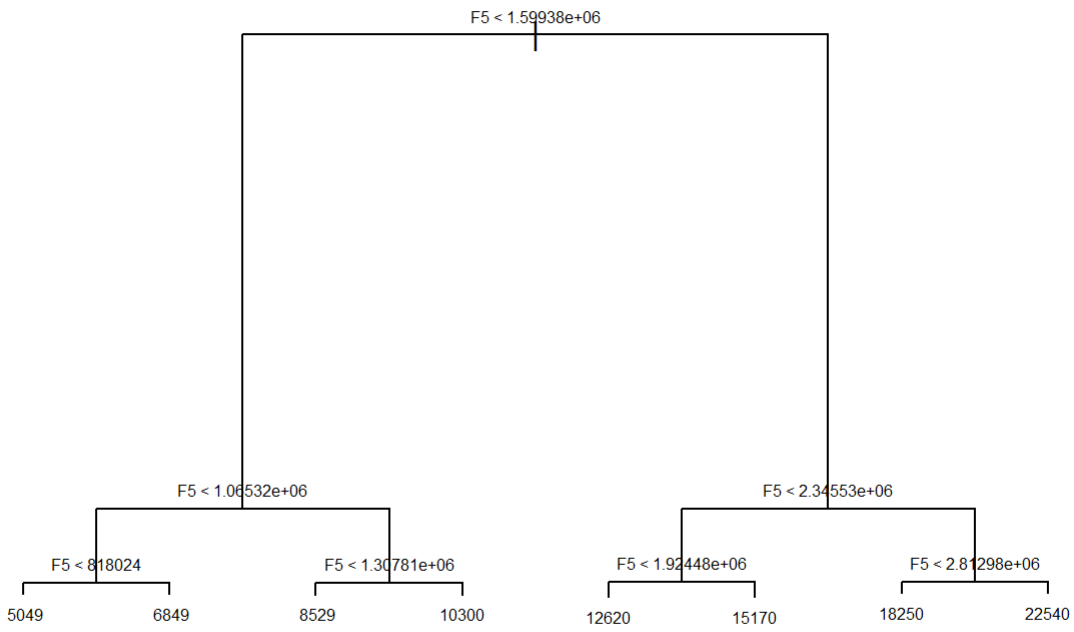
```
## [1] "mse: 592089.121641965"
```

```
print(paste('rmse:', rmse_tree))
```

```
## [1] "rmse: 769.473275456637"
```

Plotting the tree

```
plot(tree)
text(tree, cex=0.5, pretty=0)
```



Comparing results and analysis on why the results were most likely achived

Model	Correlation	rmse
Linear Regression	0.998812915661555	198.129236023442
kNN Regression	0.998008972762981	256.639383652896
Decision Tree Regression	0.981994447324154	769.473275456637

As you can see in the summary table above, the Linear Regression give the best result. It has the highest correlation and lowest rmse among 3 model. There is no surprise, as when we explore the data, we can see that, the data set is highly linear, specially with some attributes such as F2, F4, F5, F6, F9. The next ranking model is kNN Regression. Even though, kNN Regression gave a good correlation and rmse, it is not very useful for us, as it is a non-parametric model, it did not provide any coefficient. The last ranking is Decision Tree Regression, it is the because even though we instruction the model to use all the attributes as predictor, the model only pick F5 as the best information attribute. That’s why we have a bigger drop on correlation and rmse compare to kNN Regression