Lab Report 2

By: Abhishek Shah, Ravi Seth, Tanya Ralliaram

```
# Insert necessary packages
library('tidyverse')
library('gridExtra')
library('ISLR')
library('plotly')
library('caret')
library('MASS')
```

Question 1: Nonlinear Regression

1.1. Process your data

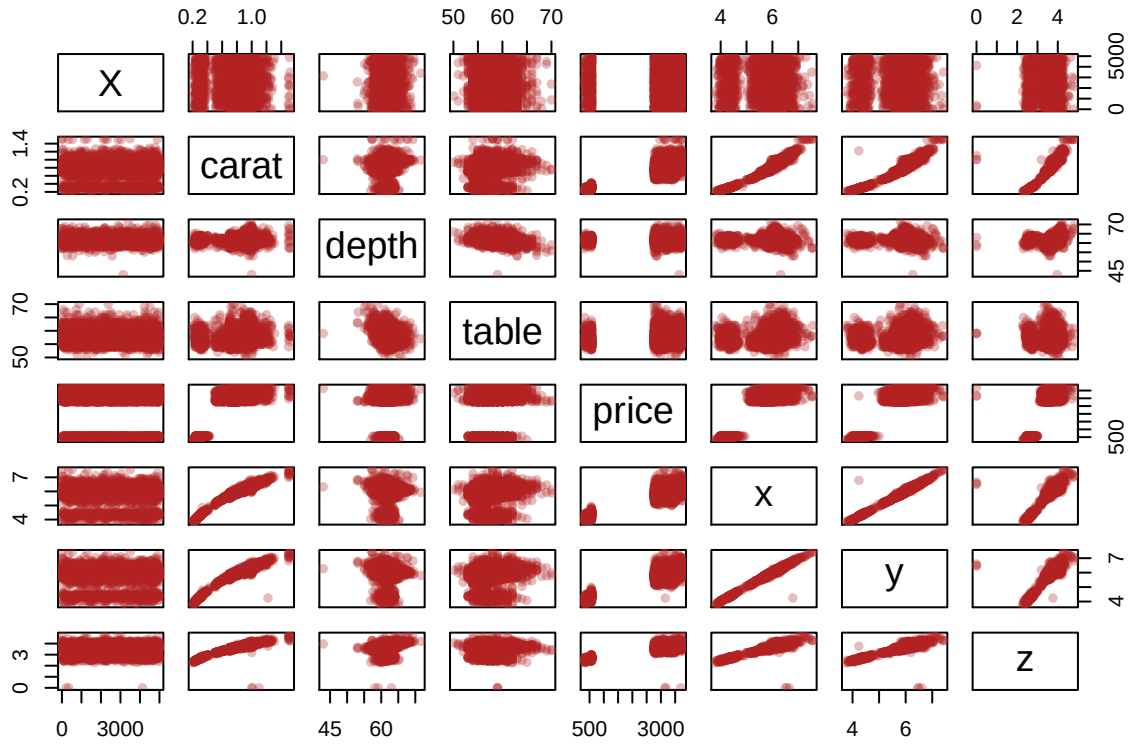
```
# Read the data
diamonds <- read.csv('data/diamonds.csv')

# Remove all rows that contain NA
diamonds <- na.omit(diamonds)

# Downsample data
diamonds <- diamonds[sample(5000), ]
dim(diamonds)
```

```
## [1] 5000   11
```

```
head(diamonds)
```

```
##         X carat       cut color clarity depth table price    x    y    z
## 110   110  0.61     Ideal     G    VVS2  60.1    57  2816 5.52 5.54 3.32
## 4516 4516  0.53     Ideal     D    VVS1  61.6    55  3097 5.21 5.27 3.23
## 2832 2832  0.30 Very Good     I     SI1  62.6    57   405 4.25 4.28 2.67
## 587   587  1.02   Premium     J     VS2  61.6    58  3513 6.40 6.35 3.93
## 1553 1553  0.73     Ideal     E     VS2  61.7    57  3470 5.75 5.78 3.55
## 782   782  0.70      Fair     G    VVS1  58.8    66  2797 5.81 5.90 3.44
```

```
numeric.cols <- summarize_all(diamonds, is.numeric) %>% unlist()
pairs(diamonds[,numeric.cols], col = adjustcolor('firebrick', .3), pch=16)
```

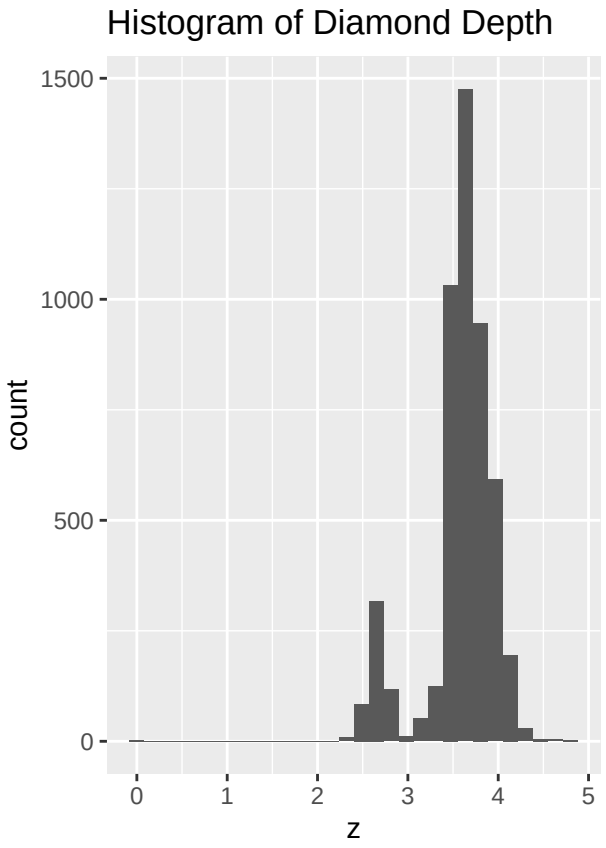


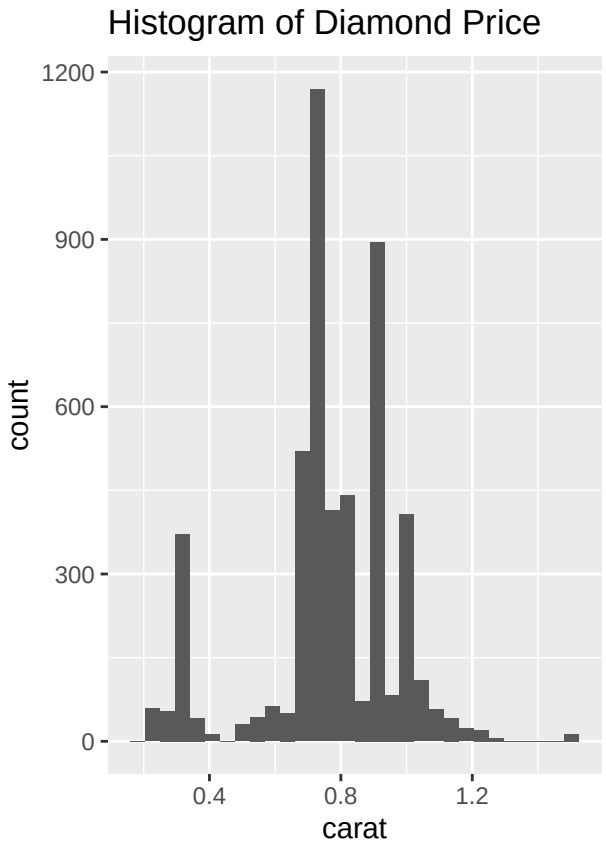
1.3. Visualize the data

It was decided to do step 1.3. before step 1.2 because before splitting the input data into train and test, it makes more sense to apply the necessary transformations first.

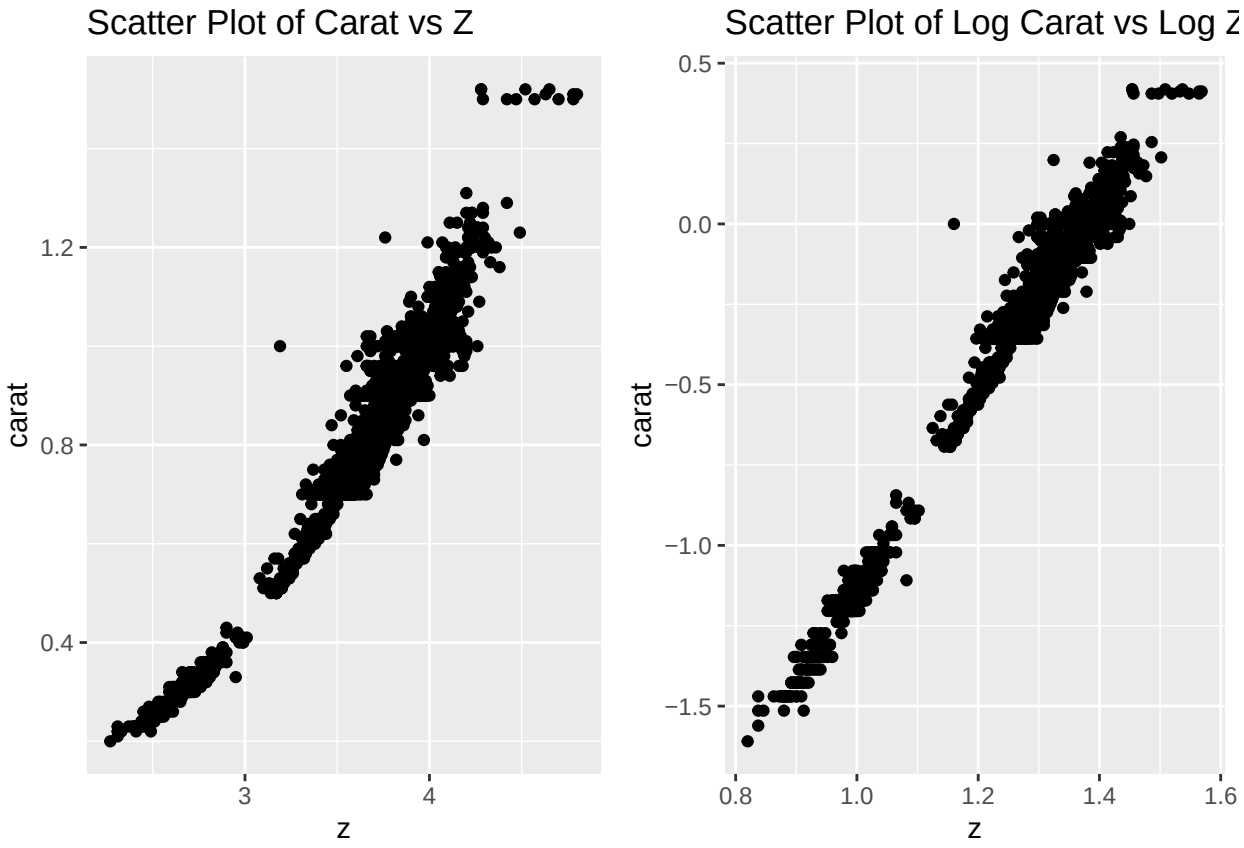
```
deadlifts <- ggplot(data = diamonds) +
  geom_histogram(aes(x = z)) +
  ggtitle("Histogram of Diamond Depth")

healthy <- ggplot(data = diamonds) +
  geom_histogram(aes(x = carat)) +
  ggtitle("Histogram of Diamond Price")

grid.arrange(deadlifts, healthy, ncol=2)
```

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






```
# Remove outliers
diamonds <- diamonds[(diamonds$z>1),]

# Apply logarithmic transform
numeric.cols <- summarize_all(diamonds, is.numeric) %>% unlist()
logDiamonds <- log(diamonds[,numeric.cols])

scatter1 <- ggplot(data = diamonds) +
  geom_point(aes(x = z, y=carat)) +
  ggtitle("Scatter Plot of Carat vs Z")

scatter2 <- ggplot(data = logDiamonds) +
  geom_point(aes(x = z, y=carat)) +
  ggtitle("Scatter Plot of Log Carat vs Log Z")

grid.arrange(scatter1, scatter2, ncol=2)
```



After removing some outliers, it can be clearly seen that as z is increased, the carat is also increased. In terms of linearity, it is slightly non-linear and showing exponential form. When applying log transform to all the carat and x values, then the relationship is clearly linear.

1.2. Train / Test Split

```
set.seed(156)
train_inds <- sample(1:nrow(logDiamonds), floor(nrow(logDiamonds)*0.8))
train <- logDiamonds[ train_inds, ]
test  <- logDiamonds[-train_inds, ]

cat('train: ', nrow(train), ', test: ', nrow(test))
```

```
## train:  3997 , test:  1000
```

1.4. Fit 4 Models

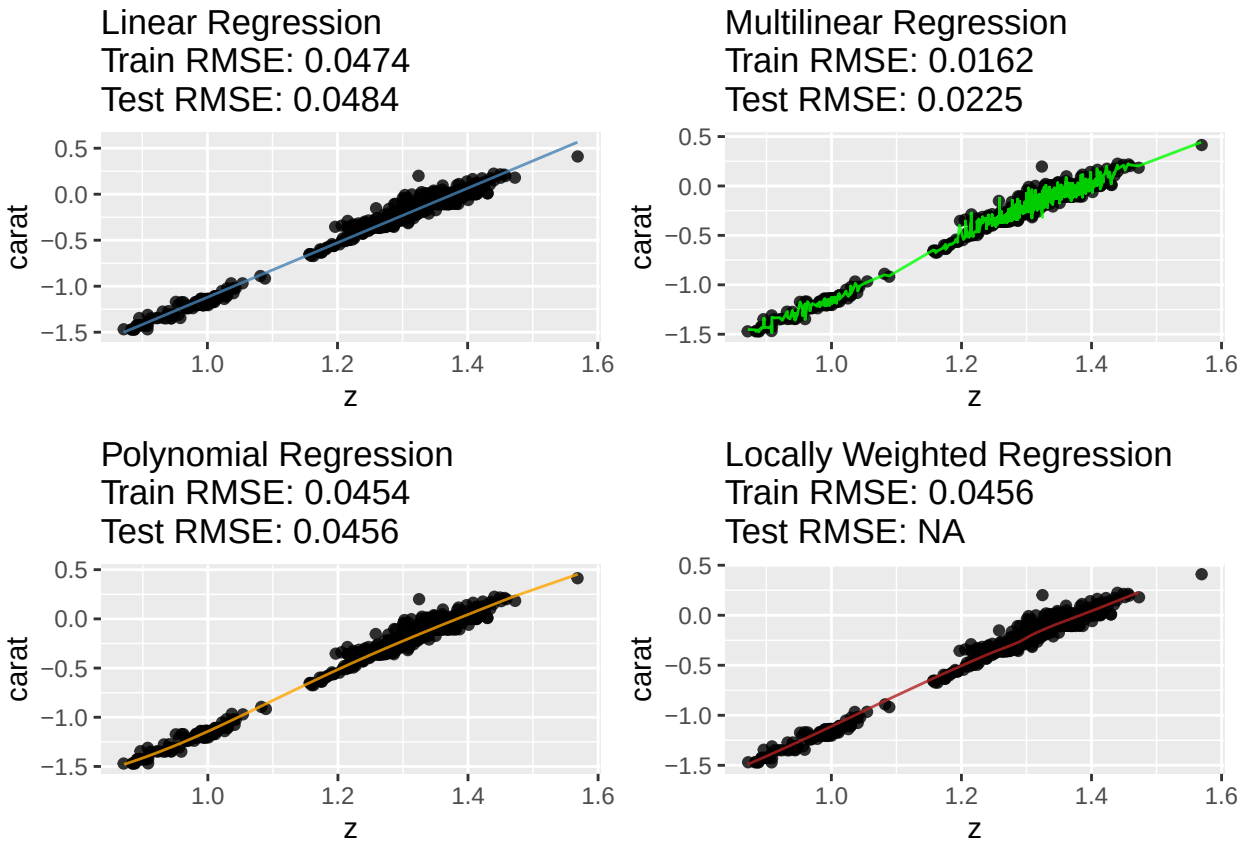
```
# Linear Regression
fit.lm  <- lm(carat ~ z, data = train)

# Predictions
preds.lm_train <- predict(fit.lm, train)
preds.lm_test <- predict(fit.lm, test)
```

```r
# RMSE
rmse.lm_train  <- RMSE(preds.lm_train, train$carat)
rmse.lm_test  <- RMSE(preds.lm_test, test$carat)

# Multilinear Regression
fit.mlm <- lm(carat ~ ., data = train)

# Predictions
preds.mlm_train <- predict(fit.mlm, train)
preds.mlm_test <- predict(fit.mlm, test)

# RMSE
rmse.mlm_train  <- RMSE(preds.mlm_train, train$carat)
rmse.mlm_test  <- RMSE(preds.mlm_test, test$carat)

# Polynomial Regression
fit.poly <- lm(carat ~ poly(z,6), data = train)

# Predictions
preds.poly_train <- predict(fit.poly, train)
preds.poly_test <- predict(fit.poly, test)

# RMSE
rmse.poly_train  <- RMSE(preds.poly_train, train$carat)
rmse.poly_test  <- RMSE(preds.poly_test, test$carat)

# Locally Weighted Regression
fit.wlm <- loess(carat ~ z, data=train)

# Predictions
preds.wlm_train <- predict(fit.wlm, train)
preds.wlm_test <- predict(fit.wlm, test)

# RMSE
rmse.wlm_train  <- RMSE(preds.wlm_train, train$carat)
rmse.wlm_test  <- RMSE(preds.wlm_test, test$carat)
```

```r
# Plots
linear_plot <- ggplot(test, aes(y=carat, x=z)) +
  geom_point(alpha=.8, position = position_jitter()) +
  geom_line(aes(y=preds.lm_test), colour = 'steelblue', alpha=.8) +
  ggtitle(paste0('Linear Regression \nTrain RMSE: ',round(rmse.lm_train, 4), "\nTest RMSE: ", rou

multilinear_plot <- ggplot(test, aes(y=carat, x=z)) +
  geom_point(alpha=.8, position = position_jitter()) +
  geom_line(aes(y=preds.mlm_test), colour = 'green', alpha=.8) +
  ggtitle(paste0('Multilinear Regression \nTrain RMSE: ',round(rmse.mlm_train, 4), "\nTest RMSE: '

poly_plot <- ggplot(test, aes(y=carat, x=z)) +
```

```
  geom_point(alpha=.8, position = position_jitter()) +
  geom_line(aes(y=preds.poly_test), colour = 'orange', alpha=.8) +
  ggtitle(paste0('Polynomial Regression \nTrain RMSE: ',round(rmse.poly_train, 4), "\nTest RMSE: '

weighted_plot <- ggplot(test, aes(y=carat, x=z)) +
  geom_point(alpha=.8, position = position_jitter()) +
  geom_line(aes(y=preds.wlm_test), colour = 'firebrick', alpha=.8) +
  ggtitle(paste0('Locally Weighted Regression \nTrain RMSE: ',round(rmse.wlm_train, 4), "\nTest RM

grid.arrange(linear_plot, multilinear_plot, poly_plot, weighted_plot, ncol=2)
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```



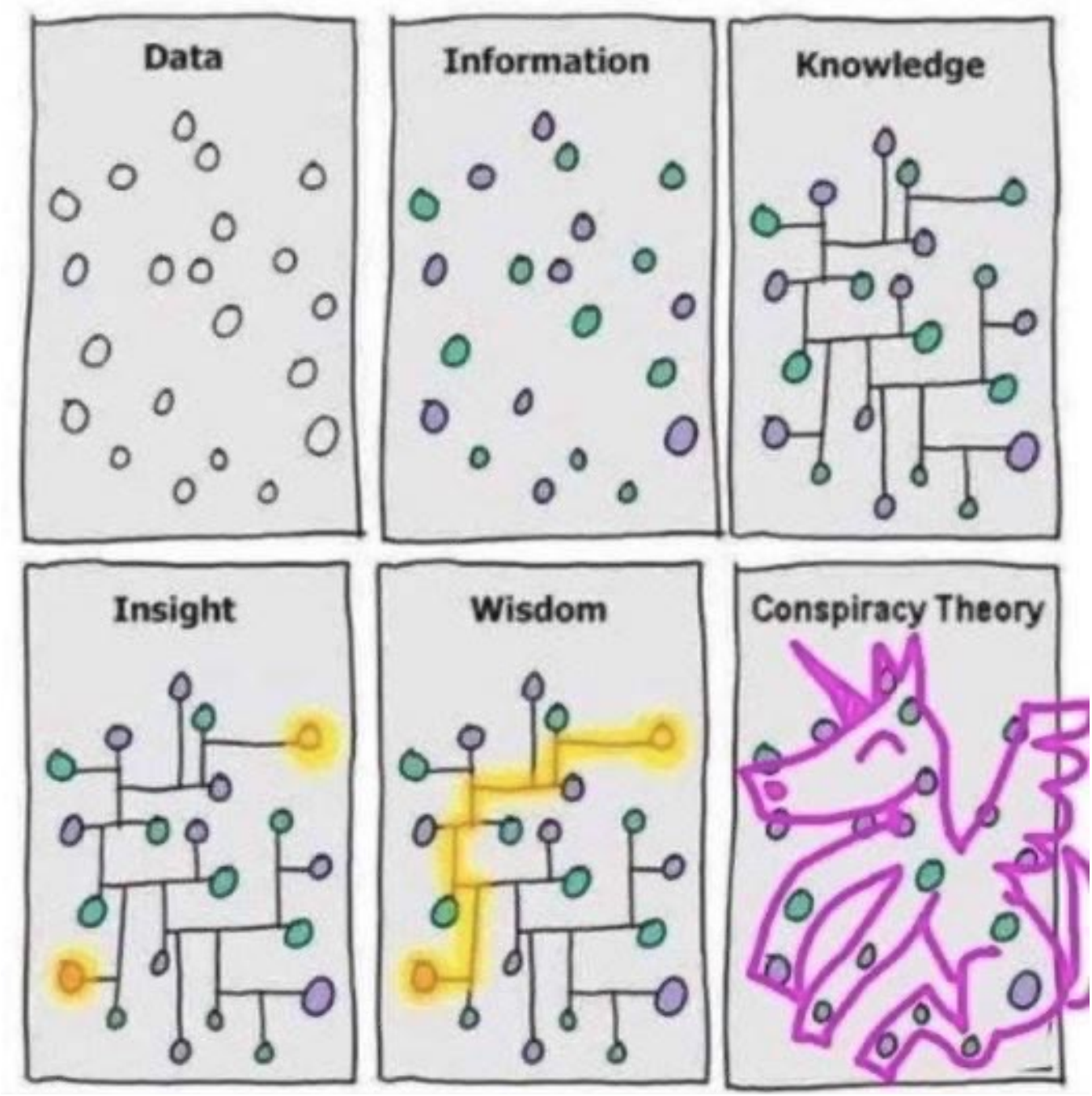
Based off of Train RMSE, the order of models from best (lowest RMSE) to worst (highest RMSE) is: 1. Multilinear Regression 2. Polynomial Regression and Locally Weighted Regression tied 3. Linear Regression

Based off of the Test RMSE, the order of models from best (lowest RMSE) to worst (highest RMSE) is: 1. Multilinear Regression 2. Polynomial Regression 3. Locally Weighted Regression 4. Linear Regression

Polynomial Regression and Locally Weighted Regression had the same training error, however when it came to test error, Polynomial Regression achieved better results.

Question 2

```
knitr::include_graphics('conspiracy.jpg')
```



Question 3

$$\theta := \theta + \frac{\alpha}{N} X^T (Y - \theta X)$$