Notes for Final

<https://www.kaggle.com/c/diamonds-datamad1021-rev/data>

The goal of this competition is the prediction of the price of diamonds based on their characteristics (weight, color, quality of cut, etc.), putting into practice all the machine learning techniques you know.

Evaluation

The evaluation metric chosen for this competition is the RMSE (Root Mean Squared Error):

<https://www.statisticshowto.com/probability-and-statistics/regression-analysis/rmse-root-mean-square-error/>

Features of the data

id: only for test & sample submission files, id for prediction sample identification

price: price in USD

carat: weight of the diamond

cut: quality of the cut (Fair, Good, Very Good, Premium, Ideal)

color: diamond colour

clarity: a measurement of how clear the diamond is

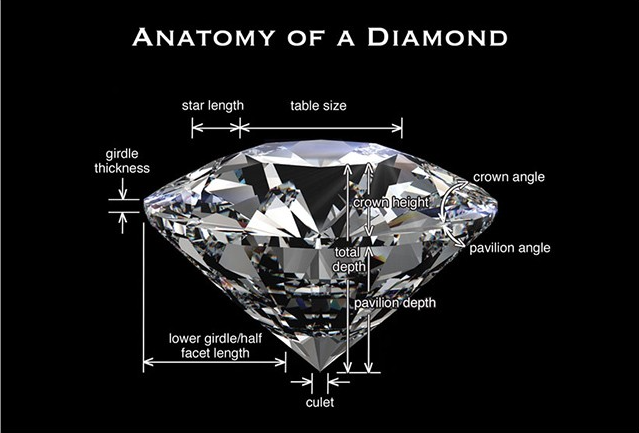
x: length in mm

y: width in mm

z: depth in mm

depth: total depth percentage = z / mean(x, y) = 2 \* z / (x + y) (43--79)

table: width of top of diamond relative to widest point (43--95)



Running list of packages and libraries I use and add any others I might need:

install.packages("caret")

library(caret)

library(dplyr)

install.packages("tidyverse")

library(tidyverse)

install.packages("cluster")

library(cluster)

install.packages("factoextra")

library(factoextra)

install.packages("cowplot")

library(cowplot)

library(ggplot2)

install.packages("tidyr")

library(tidyr)

library(dplyr)

install.packages("tidyverse")

library(tidyverse)

install.packages("cluster")

library(cluster)

library(readr)

library(tidyr)

install.packages("devtools")

library(devtools)

library(cluster)

install.packages("fpc")

library(fpc)

library(readr)

library(dplyr)

library(ggplot2)

install.packages("ggcorrplot")

library(ggcorrplot)

library(tidyr)

library(fastDummies)

library(caret)

Code notes:

Apply past learned codes to this project to see how they work

glimpse(train)

summary(train)

converting character variables to factors

train %>% mutate(cut = as.factor(cut), color = as.factor(color), clarity <- as.factor(clarity))

bar plots

ggplot(train, aes(x=cut, fill = cut)) + geom\_bar() + theme\_classic() + labs(title="Various types of diamond cuts", x="Cut categories", y = "Count")

ggplot(train, aes(x=clarity, fill = clarity)) + geom\_bar() + theme\_classic() + labs(title="Various types of diamond clarity levels", x="diamond clarity levels", y = "Count")

histogram plots

ggplot(train, aes(x = depth)) + geom\_histogram(fill = 'blue', bins=100) + labs(x="depth", y="Count",title = "Probability Distribution of depth") + theme\_classic()

ggplot(train, aes(x = log(carat))) + geom\_histogram(fill = 'blue', bins=100) + labs(x="carat", y="Count",title = "Probability Distribution of carat") + theme\_classic()

is.na?

apply(train,2,function(x){any(is.na(x))})

is there any correlation to the variables?

train\_cor <- round(cor(train %>% select\_if(is.numeric)), 1)

ggcorrplot(train\_cor, title = "Correlation", type = "lower") + theme(plot.title = element\_text(hjust = 0.5), axis.text.x = element\_text(angle = 90))

x, y, z is very correlated to each other and also it's very correlated with caret variable --- remove from dataset

train <- train %>% select(-c(x,y,z))

box plot for all numeric variables

train %>% select\_if(is.numeric) %>% mutate\_all(scale) %>% gather("features","values") %>% na.omit() %>%

ggplot(aes(x = features, y = values)) + geom\_boxplot(show.legend = FALSE) + stat\_summary(fun = mean, geom = "point", pch = 1) +

scale\_y\_continuous(name = "Variable values", minor\_breaks = NULL) + scale\_fill\_brewer(palette = "Set1") + coord\_flip() + theme\_minimal() + labs(x = "Variable names") + ggtitle(label = "Distribution of numeric variables in diamond train dataset")

creating a dummy

train\_d <- dummy\_cols(train)

train\_d <- train\_d %>% select(-c(cut, color, clarity))

Splitting dataset into training (60%) and validation (40%) sets

set.seed(23)

index <- createDataPartition(train\_d$price, p=0.6, list = FALSE)

train\_df <- train\_d[index,]

test\_df <- train\_d[-index,]

Defining a function to normalize the data

scale\_fun <- preProcess(train\_df %>% select(-price), method = c("center", "scale"))

train\_norm <- predict(scale\_fun, train\_df)

test\_norm <- predict(scale\_fun, test\_df)

Building a model to estimate the diamond price value

diamond\_train\_model <- lm(price ~ . , data = train\_norm)

Performance metrics on test data

RMSE on test data

(linear\_base\_rsme <- sqrt( mean(( test\_norm$price - predict( diamond\_train\_model, test\_norm))^2)))

R squared on test data

(linear\_base\_rsquare <- cor( test\_norm$price, predict( diamond\_train\_model, test\_norm))^2)