

# XL\_Final

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed      dist
##  Min.   : 4.0    Min.   :  2.00
##  1st Qu.:12.0    1st Qu.: 26.00
##  Median :15.0    Median : 36.00
##  Mean   :15.4    Mean   : 42.98
##  3rd Qu.:19.0    3rd Qu.: 56.00
##  Max.   :25.0    Max.   :120.00
```

## Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

```
install.packages("caret") library(caret) library(dplyr) install.packages("tidyverse") library(tidyverse) in-
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) li-
brary(ggplot2) install.packages("ggcorrplot") library(ggcorrplot) library(tidyr) library(fastDummies)
library(caret)

setwd("C:/Users/xlamo/Desktop/ML64060XLamoreux")

library(readr)

train <- read_csv("XL_Final/train.csv")
test <- read_csv("XL_Final/test.csv")

summary(test) summary(train)

head(test) head(train)

glimpse(test) glimpse(train)

diamond_test <- read_csv("XL_Final/test.csv") diamond_train <- read_csv("XL_Final/train.csv")

head(diamond_test) summary(diamond_train)

sum(is.na(diamond_test)) sum(is.na(diamond_train))
```

### converting character variables to factors

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

### bar plot on cut variable

```
ggplot(train, aes(x=cut, fill = cut)) + geom_bar() + theme_classic() + labs(title="Various types of diamond cuts", x="Cut categories", y = "Count")
```

### bar plot on clarity variable

```
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")
```

### Checking the distribution of depth column

```
ggplot(train, aes(x = depth)) + geom_histogram(fill = 'blue', bins=100) + labs(x="depth", y="Count", title = "Probability Distribution of depth") + theme_classic()
```

### Checking the distribution of carat column

```
ggplot(train, aes(x = log(carat))) + geom_histogram(fill = 'blue', bins=100) + labs(x="carat", y="Count", title = "Probability Distribution of carat") + theme_classic()
apply(train, 2, function(x){any(is.na(x))})
```

### Correlation to answer how these variables are related to the price

```
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))
```

**since x, y, z is highly correlated to each other and also it's highly correlated with caret variable, so removing 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) +
```

### Add average to the boxplot

```
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")
```

### Converting category variable to numeric variable.

```
train_d <- dummy_cols(train) train_d <- train_d %>% select(-c(cut, color, clarity)) View(train_d)
```

### 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)
```

### Summary statistics of normalized data

```
summary(train_norm)
```

### Building a model to estimate the diamond price value

```
diamond_train_model <- lm(price ~ . , data = train_norm) summary(diamond_train_model)
```

### Performance metrics on test data

#### RMSE on test data

```
(linear_base_rsme <- sqrt( mean(( test_norm$price - predict( diamond_train_model, test_norm))^2)))
```

**linear base rmse is 1178.782** — 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/>

#### R squared on test data

```
(linear_base_rsquare <- cor( test_norm$price, predict( diamond_train_model, test_norm))^2)
```

**r squared is 0.9137193** predicting a **91.37%** accuracy of the model when applying this model to predict diamond prices based on the variables color, cut, clarity, carat, length, width, depth, and table of the diamond.

```
summary(test) summary(test_norm) view(test_norm)
```

data needed a lot of transformation before predicting the model and the model is giving an accuracy of 91%