title: "Machine Learning: Predicting the Cut of a Diamond" author: "Arya Shah" ---

```
library(neuralnet)
```

```
## Warning: package 'neuralnet' was built under R version 4.1.2
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.2
```

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

```
## Warning: package 'ggplot2' was built under R version 4.1.2
```

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

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.1.2
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.0     v forcats 0.5.1
## v purrr   0.3.4
```

```
## Warning: package 'tibble' was built under R version 4.1.2
```

```
## Warning: package 'tidyr' was built under R version 4.1.2
```

```
## Warning: package 'readr' was built under R version 4.1.2
```

```
## Warning: package 'purrr' was built under R version 4.1.2
```

```
## Warning: package 'stringr' was built under R version 4.1.2
```

```
## Warning: package 'forcats' was built under R version 4.1.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::compute() masks neuralnet::compute()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x purrr::lift()    masks caret::lift()
```

```
?diamonds
```

```
## starting httpd help server ... done
```

```
head(diamonds)
```

```
## # A tibble: 6 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
## 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
## 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
## 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
## 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
## 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

```
df <- diamonds%>%
  filter(cut %in% c("Ideal","Good"))
```

```
dim(df)
```

```
## [1] 26457    10
```

```
df$binary <- ifelse(df$cut == "Ideal", 1, 0)
head(df)
```

```
## # A tibble: 6 x 11
##   carat cut   color clarity depth table price     x     y     z binary
##   <dbl> <ord> <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>  <dbl>
## 1  0.23 Ideal E     SI2      61.5    55   326  3.95  3.98  2.43      1
## 2  0.23 Good  E     VS1      56.9    65   327  4.05  4.07  2.31      0
## 3  0.31 Good  J     SI2      63.3    58   335  4.34  4.35  2.75      0
## 4  0.3  Good  J     SI1      64      55   339  4.25  4.28  2.73      0
## 5  0.23 Ideal J     VS1      62.8    56   340  3.93  3.9   2.46      1
## 6  0.31 Ideal J     SI2      62.2    54   344  4.35  4.37  2.71      1
```

```
df$binary <- as.factor(df$binary)
names(df)
```

```
##  [1] "carat"   "cut"     "color"   "clarity" "depth"   "table"   "price"
##  [8] "x"       "y"       "z"       "binary"
```

```
df <- df[ , -2]
head(df)
```

```
## # A tibble: 6 x 10
##   carat color clarity depth table price     x     y     z binary
##   <dbl> <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl> <fct>
## 1  0.23 E     SI2      61.5    55   326  3.95  3.98  2.43 1
## 2  0.23 E     VS1      56.9    65   327  4.05  4.07  2.31 0
## 3  0.31 J     SI2      63.3    58   335  4.34  4.35  2.75 0
## 4  0.3  J     SI1      64      55   339  4.25  4.28  2.73 0
## 5  0.23 J     VS1      62.8    56   340  3.93  3.9   2.46 1
## 6  0.31 J     SI2      62.2    54   344  4.35  4.37  2.71 1
```

```
rows <- createDataPartition(df$binary, p=.8, list=F, times=1)
train <- df[rows,]
head(train)
```

```
## # A tibble: 6 x 10
##   carat color clarity depth table price     x     y     z binary
##   <dbl> <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl> <fct>
## 1  0.23 E     SI2      61.5    55   326  3.95  3.98  2.43 1
## 2  0.23 E     VS1      56.9    65   327  4.05  4.07  2.31 0
## 3  0.31 J     SI2      63.3    58   335  4.34  4.35  2.75 0
## 4  0.3  J     SI1      64      55   339  4.25  4.28  2.73 0
## 5  0.23 J     VS1      62.8    56   340  3.93  3.9   2.46 1
## 6  0.31 J     SI2      62.2    54   344  4.35  4.37  2.71 1
```

```
test <- df[-rows,]
```

```
control <- trainControl(method="repeatedcv", number=2, repeats=2)

model <- train(binary ~ ., data=train, method="ranger", trControl=control)
```

```
model
```

```
## Random Forest
##
## 21166 samples
##     9 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (2 fold, repeated 2 times)
## Summary of sample sizes: 10583, 10583, 10583, 10583
## Resampling results across tuning parameters:
##
##   mtry  splitrule   Accuracy   Kappa
##    2    gini        0.9764481  0.9200664
##    2    extratrees  0.8189162  0.1993338
##   11    gini        0.9825428  0.9420936
##   11    extratrees  0.9811726  0.9372410
##   20    gini        0.9818105  0.9396756
##   20    extratrees  0.9819522  0.9400000
##
## Tuning parameter 'min.node.size' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 11, splitrule = gini
##  and min.node.size = 1.
```

#Trying with 5 folds, repeated 5 times

```
control1 <- trainControl(method="repeatedcv", number=5, repeats=5)

model1 <- train(binary ~ ., data=train, method="ranger", trControl=control1)

model1
```

```
## Random Forest
##
## 21166 samples
##     9 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 16933, 16933, 16932, 16933, 16933, 16933, ...
## Resampling results across tuning parameters:
##
##   mtry  splitrule   Accuracy   Kappa
##    2    gini        0.9783899  0.9269600
##    2    extratrees  0.8328074  0.1521830
##   11    gini        0.9825097  0.9419663
##   11    extratrees  0.9814987  0.9383946
##   20    gini        0.9820751  0.9405489
##   20    extratrees  0.9816309  0.9389745
##
## Tuning parameter 'min.node.size' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 11, splitrule = gini
##  and min.node.size = 1.
```

```
pred <- predict(model,test)
confusionMatrix(pred,test$binary)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  933   27
##          1   48 4283
##
##                Accuracy : 0.9858
##                  95% CI : (0.9823, 0.9888)
##     No Information Rate : 0.8146
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9527
##
##  Mcnemar's Test P-Value : 0.02092
##
##             Sensitivity : 0.9511
##             Specificity : 0.9937
##          Pos Pred Value : 0.9719
##          Neg Pred Value : 0.9889
##              Prevalence : 0.1854
##          Detection Rate : 0.1763
##    Detection Prevalence : 0.1814
##       Balanced Accuracy : 0.9724
##
##        'Positive' Class : 0
##
```

```
pred1 <- predict(model1,test)
confusionMatrix(pred1,test$binary)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  933   27
##          1   48 4283
##
##                Accuracy : 0.9858
##                  95% CI : (0.9823, 0.9888)
##     No Information Rate : 0.8146
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9527
##
##  Mcnemar's Test P-Value : 0.02092
##
##             Sensitivity : 0.9511
##             Specificity : 0.9937
##          Pos Pred Value : 0.9719
##          Neg Pred Value : 0.9889
##              Prevalence : 0.1854
##          Detection Rate : 0.1763
##    Detection Prevalence : 0.1814
##       Balanced Accuracy : 0.9724
##
##        'Positive' Class : 0
##
```