ML_model

- Data detail : https://bookdown.org/yih_huynh/Guide-to-R-Book/diamonds.html

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0     v purrr   1.0.1
## v tibble  3.1.8     v dplyr   1.1.0
## v tidyr   1.3.0     v stringr 1.5.0
## v readr   2.1.3     v forcats 1.0.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
tibble(diamonds)
```

```
## # A tibble: 53,940 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
##  7  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47
##  8  0.26 Very Good H     SI1      61.9    55   337  4.07  4.11  2.53
##  9  0.22 Fair      E     VS2      65.1    61   337  3.87  3.78  2.49
## 10  0.23 Very Good H     VS1      59.4    61   338  4     4.05  2.39
## # ... with 53,930 more rows
```

```
# check null
mean(complete.cases(diamonds))
```

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

```
# train_test_split
train_test_data = function(data, train_size=0.7) {
  set.seed(7)
```

```r
  n = nrow(data)
  id = sample(1:n, size = n*train_size)
  train_data = data[id, ]
  test_data = data[-id, ]
  return(list(train_data, test_data))
}

split_data = train_test_data(diamonds)
train_data = split_data[[1]]
nrow(train_data)
```

```
## [1] 37758
```

```r
test_data = split_data[[2]]
nrow(test_data)
```

```
## [1] 16182
```

Linear regression model

```r
lm_model =
train(price ~ .,
      data = train_data,
      method="lm")
lm_model
```

```
## Linear Regression
##
## 37758 samples
##     9 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 37758, 37758, 37758, 37758, 37758, 37758, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   1128.136  0.9197066  741.9278
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
# score + evaluate model (Linear regression)
p1 = predict(lm_model, newdata= test_data)
RMSE(p1, test_data$price)
```

```
## [1] 1245.287
```

set K fold cross validation

```
ctrl = trainControl(method = "cv",
                    number = 5,
                    verboseIter = T)
```

Logistic Regression model

```
glm_model =
train(price~ .,
      data = train_data,
      method="glm",
      trControl = ctrl)
```

```
## + Fold1: parameter=none
## - Fold1: parameter=none
## + Fold2: parameter=none
## - Fold2: parameter=none
## + Fold3: parameter=none
## - Fold3: parameter=none
## + Fold4: parameter=none
## - Fold4: parameter=none
## + Fold5: parameter=none
## - Fold5: parameter=none
## Aggregating results
## Fitting final model on full training set
```

```
glm_model
```

```
## Generalized Linear Model
##
## 37758 samples
##     9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 30207, 30205, 30207, 30207, 30206
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   1129.125  0.9195944  741.9947
```

```
# score + evaluate model (Logistic Regression)
p2 = predict(glm_model, newdata= test_data)
RMSE(p2, test_data$price)
```

```
## [1] 1245.287
```

set grid search

```
grid = data.frame(k = c(3,7))
```

KNN model

```
knn_model =
train(price~.,
      data= train_data,
      method = "knn",
      trControl = ctrl,
      tuneGrid = grid)
```

```
## + Fold1: k=3
## - Fold1: k=3
## + Fold1: k=7
## - Fold1: k=7
## + Fold2: k=3
## - Fold2: k=3
## + Fold2: k=7
## - Fold2: k=7
## + Fold3: k=3
## - Fold3: k=3
## + Fold3: k=7
## - Fold3: k=7
## + Fold4: k=3
## - Fold4: k=3
## + Fold4: k=7
## - Fold4: k=7
## + Fold5: k=3
## - Fold5: k=3
## + Fold5: k=7
## - Fold5: k=7
## Aggregating results
## Selecting tuning parameters
## Fitting k = 7 on full training set
```

```
knn_model
```

```
## k-Nearest Neighbors
##
## 37758 samples
##     9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 30206, 30205, 30207, 30207, 30207
## Resampling results across tuning parameters:
##
```

```
##   k  RMSE      Rsquared   MAE
##   3  1024.520  0.9347872  547.7812
##   7  1012.071  0.9391815  540.8551
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 7.
```

```r
# score + evaluate model (KNN)
p3 = predict(knn_model, newdata= test_data)
RMSE(p3, test_data$price)
```

```
## [1] 981.4582
```