PROJECT 1

1. Cleaning Data

```r
library(rpart)
library(rpart.plot)
library(forecast)
```

```
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo
```

```r
library(caret)
```

```
Loading required package: ggplot2

Loading required package: lattice
```

```r
library(ROSE)
```

```
Loaded ROSE 0.0-4
```

```r
library(car)
```

```
Loading required package: carData
```

```r
library(dbplyr)
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:dbplyr':

    ident, sql

The following object is masked from 'package:car':

    recode

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
cars <- read.csv('car_train_class_5.csv', header=TRUE)
```

```r
cars <- cars[, -c(2)]
cars <- cars[,c(3,7,10,14,15,18:22,24:27,33,34,36,51,52,55:58)]
t(t(names(cars)))
```

```
      [,1]
 [1,] "back_legroom"
 [2,] "daysonmarket"
 [3,] "engine_displacement"
 [4,] "frame_damaged"
 [5,] "franchise_dealer"
 [6,] "fuel_tank_volume"
 [7,] "fuel_type"
 [8,] "has_accidents"
 [9,] "height"
```

```
[10,] "horsepower"
[11,] "is_cpo"
[12,] "is_new"
[13,] "is_oemcpo"
[14,] "length"
[15,] "maximum_seating"
[16,] "mileage"
[17,] "owner_count"
[18,] "wheelbase"
[19,] "width"
[20,] "power_rpm"
[21,] "torque_lbft"
[22,] "torque_rpm"
[23,] "price_nom"
```

```r
# This is to remove the na (but not remove the empty string)
# The reason empty string exist is similar to reality that some people just lazy and don't
cars <- na.omit(cars)
```

```r
cars[cars == ""] <- "Missing"
```

```r
cars$price_nom <- factor(cars$price_nom, levels = c('0', '1'), labels = c('low', 'high'))
```

```r
factor_columns <- c('frame_damaged', 'franchise_dealer', 'fuel_type', 'has_accidents', 'is

cars[factor_columns] <- lapply(cars[factor_columns], factor)
```

2. KNN Model

```r
set.seed(777)
train_index <- sample(1:nrow(cars), 0.7 * nrow(cars))
valid_index <- setdiff(1:nrow(cars), train_index)
```

```r
train_df <- cars[train_index, ]
valid_df <- cars[valid_index, ]
```

```r
unique(cars$price_nom)
```

```
[1] high low
Levels: low high
```

```r
head(cars)
```

```
   back_legroom daysonmarket engine_displacement frame_damaged franchise_dealer
4          43.6           48                5000       Missing             True
6          36.5           62                3600         False            False
7          40.9            5                6200         False             True
8          39.5           42                3500         False             True
9          39.5            6                3500         False            False
10         32.0           18                4000         False             True
   fuel_tank_volume         fuel_type has_accidents height horsepower is_cpo
4              26.0 Flex Fuel Vehicle       Missing   77.2        395  False
6              20.0 Flex Fuel Vehicle         False   67.9        283  False
7              26.0          Gasoline         False   74.0        420  False
8              18.6          Gasoline         False   70.0        290  False
9              18.6          Gasoline         False   71.0        290  False
10             17.4          Gasoline         False   55.2        469   True
   is_new is_oemcpo length maximum_seating mileage owner_count wheelbase width
4    True     False  231.9               6       7           0     145.0  96.8
6   False     False  202.8               7  109898           1     121.2  88.5
7   False     False  230.0               6   43027           1     143.5  80.0
8   False     False  198.3               7   21289           1     112.8  90.2
9   False     False  198.3               7   38120           1     112.8  90.2
10  False     False  187.0               4   16145           2     111.8  79.3
   power_rpm torque_lbft torque_rpm price_nom
4       5750         400       4500      high
6       6400         260       4400       low
7       5600         383       4100       low
8       6500         255       4000       low
9       6500         255       4000       low
10      5500         479       1750      high
```

```r
table(cars$price_nom)
```

```

  low  high 
19358  2659 
```

2.1. Normalization

```
train_norm <- train_df
valid_norm <- valid_df
names(train_df)
```

```
 [1] "back_legroom"        "daysonmarket"        "engine_displacement"
 [4] "frame_damaged"       "franchise_dealer"    "fuel_tank_volume"
 [7] "fuel_type"           "has_accidents"       "height"
[10] "horsepower"          "is_cpo"              "is_new"
[13] "is_oemcpo"           "length"              "maximum_seating"
[16] "mileage"             "owner_count"         "wheelbase"
[19] "width"               "power_rpm"           "torque_lbft"
[22] "torque_rpm"          "price_nom"
```

```
norm_values <- preProcess(train_df[, -c(23)],
                          method = c("center",
                                     "scale"))
train_norm[, -c(23)] <- predict(norm_values,
                                train_df[, -c(23)])

head(train_norm)
```

```
      back_legroom daysonmarket engine_displacement frame_damaged
16383    0.4745129   -0.3340900            0.431850         False
17027    0.1041721    0.7884744            0.431850         False
933      0.7214068   -0.6004612            0.431850         False
15028   -0.1118600   -0.3245767            0.431850         False
21732    0.4436512   -0.3436032            0.204393         False
20324    0.1967573   -0.6004612            0.431850       Missing
      franchise_dealer fuel_tank_volume         fuel_type has_accidents
16383            False      0.127144665          Gasoline         False
17027             True      0.646907637          Gasoline         False
933               True      1.446542978 Flex Fuel Vehicle         False
15028             True     -0.592527141           Missing         False
21732             True      0.007199364          Gasoline         False
20324             True      1.166670609          Gasoline       Missing
            height horsepower is_cpo is_new is_oemcpo      length
16383  0.009217659  0.6847298  False  False     False -0.22210625
17027  1.300090088  0.3983827  False   True     False  1.23904582
```

```
933    1.802096032  0.6274604  False  False     False  1.80299925
15028 -1.037823311  0.5129216  False  False     False -0.07855447
21732  0.080932794  0.4556522  False  False     False -0.32977009
20324  0.482537549  0.5129216  False   True     False -0.20672570
      maximum_seating     mileage owner_count   wheelbase      width  power_rpm
16383      -0.5188231 -0.23583556   0.1597655 -0.19921252 -0.4503343  1.3022523
17027      -0.5188231 -0.73217916  -0.8415141  1.56295435 -0.6158047  0.8606806
933         0.3599089  0.39597440   0.1597655  1.79033072  0.1563903 -0.3168442
15028      -0.5188231  0.26121143   0.1597655 -0.51896054 -0.6433831  0.7870853
21732       1.2386409  0.04846134   0.1597655 -0.41948338 -0.5330695  0.8606806
20324      -0.5188231 -0.73400355  -0.8415141 -0.04289127  0.9010069  0.8606806
       torque_lbft torque_rpm price_nom
16383  0.077203673  1.1281760       low
17027 -0.018708121  1.0411466      high
933    1.289179971  0.3013965       low
15028 -0.001269613  0.5624848       low
21732 -0.088462152  1.4762937       low
20324 -0.018708121  0.3449112       low
```

```r
valid_norm[, -c(23)] <- predict(norm_values,
                                valid_df[, -c(23)])

head(valid_norm)
```

```
   back_legroom daysonmarket engine_displacement frame_damaged franchise_dealer
15   0.01158692   -0.6480275         -0.40215899         False             True
18  -1.03771199    0.4364839         -1.08452995       Missing             True
20   0.04244865   -0.5719214         -0.40215899         False             True
25  -0.48220081    2.0442244         -0.02306401         False             True
30   0.32020424    1.3592699          2.40314386       Missing             True
33   0.13503385   -0.3626298          0.35603097         False             True
   fuel_tank_volume fuel_type has_accidents       height horsepower is_cpo
15      -0.87239951  Gasoline         False -1.267311743 -0.5522896  False
18      -1.11229011  Gasoline       Missing -0.621875529 -0.8615445  False
20      -0.05277329  Gasoline         False -0.994794230 -0.8615445  False
25      -0.31265477  Gasoline         False  0.009217659  1.5094095  False
30       1.44654298  Gasoline       Missing  1.214031926  1.9446570  False
33       1.16667061  Gasoline         False  0.683339927  0.5930988  False
   is_new is_oemcpo      length maximum_seating    mileage owner_count
15   True     False -0.08880817      -0.5188231 -0.7339816  -0.8415141
18   True     False -1.52945284      -0.5188231 -0.7338277  -0.8415141
```

```
20  False     False -0.24261365      -0.5188231 -0.4450028   0.1597655
25  False     False -0.39641913      -0.5188231 -0.7120449  -0.8415141
30   True     False  0.51616006       2.1173730 -0.7339376  -0.8415141
33  False     False -0.21185255      -0.5188231  0.0892573   0.1597655
     wheelbase      width   power_rpm torque_lbft  torque_rpm price_nom
15 -0.29158417 -0.8088534  1.15506175 -0.69880993  0.69302890       low
18 -0.91686919 -1.0156914 -0.46403480 -0.58545963 -1.83082442       low
20 -0.49764401  0.4183850 -0.02246302 -0.76856396  0.34491120       low
25 -0.17789599  0.8182717 -0.02246302  0.92297131 -1.81341853      high
30  0.04948038  0.3080715 -0.31684421  1.72514267  0.43194063      high
33 -0.03578576  0.8320609  1.00787115  0.09464218 -0.09023592       low
```

```r
train_norm <- mutate_if(train_norm, is.character, as.factor)
```

```r
knn_model <- caret::knn3(price_nom ~ ., data = train_norm, k = 5)
knn_model
```

```
5-nearest neighbor model
Training set outcome distribution:

  low  high
13535  1876
```

```r
knn_pred_train <- predict(knn_model, newdata = train_norm[, -c(23)], type = "class")
head(knn_pred_train)
```

```
[1] low  high low  low  low  low
Levels: low high
```

```r
confusionMatrix(knn_pred_train, as.factor(train_norm[, 23]), positive = "high") # k =5
```

```
Confusion Matrix and Statistics

          Reference
Prediction   low  high
      low  13261   377
      high   274  1499
```

```
                Accuracy : 0.9578
                  95% CI : (0.9545, 0.9609)
     No Information Rate : 0.8783
     P-Value [Acc > NIR] : < 2.2e-16

                   Kappa : 0.7977

  Mcnemar's Test P-Value : 6.396e-05

             Sensitivity : 0.79904
             Specificity : 0.97976
          Pos Pred Value : 0.84546
          Neg Pred Value : 0.97236
              Prevalence : 0.12173
          Detection Rate : 0.09727
    Detection Prevalence : 0.11505
       Balanced Accuracy : 0.88940

        'Positive' Class : high
```

```r
knn_pred_valid <- predict(knn_model, newdata = valid_norm[, -c(23)], type = "class")
head(knn_pred_valid)
```

```
[1] low  low  low  low  high low
Levels: low high
```

```r
confusionMatrix(knn_pred_valid, as.factor(valid_norm[, 23]), positive = "high") # k =5
```

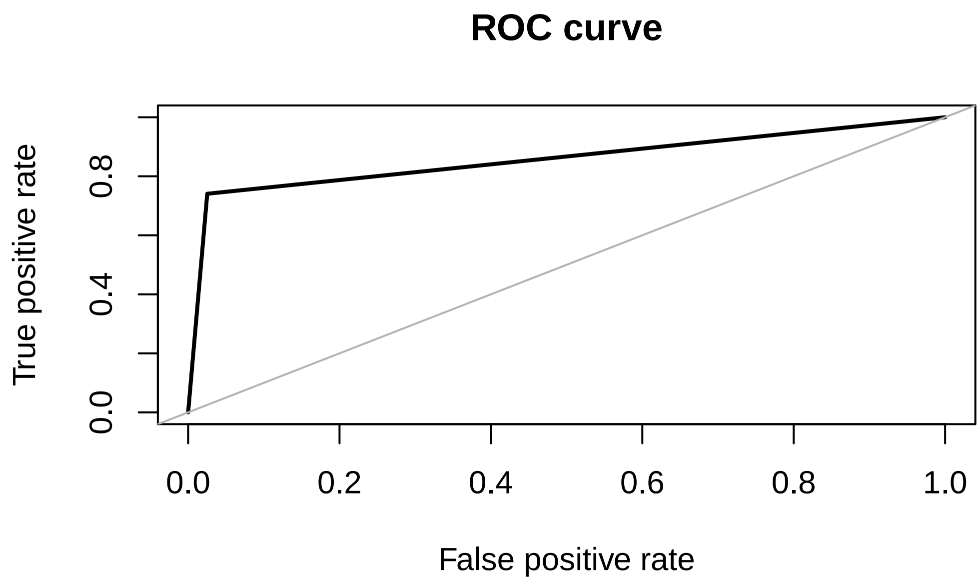
```
Confusion Matrix and Statistics

          Reference
Prediction  low high
      low  5676  203
      high  147  580

                Accuracy : 0.947
                  95% CI : (0.9413, 0.9523)
     No Information Rate : 0.8815
     P-Value [Acc > NIR] : < 2.2e-16
```

```
                 Kappa : 0.7383

 Mcnemar's Test P-Value : 0.003283

           Sensitivity : 0.7407
           Specificity : 0.9748
        Pos Pred Value : 0.7978
        Neg Pred Value : 0.9655
            Prevalence : 0.1185
        Detection Rate : 0.0878
  Detection Prevalence : 0.1101
     Balanced Accuracy : 0.8577

      'Positive' Class : high
```

2.2. ROC curve

The ROC AUC score shows how well the classifier distinguishes positive and negative classes. It can take value from 0 to 1. A higher ROC AUC indicates a better performance.

```
ROSE::roc.curve(valid_norm$price_nom, knn_pred_valid)
```




```
Area under the curve (AUC): 0.858
```

Note: The reason I keep this is that the down side of kNN is that it **CANNOT** predict if there is a lack in variable in any row in the data set.

3. Decision Tree Model

NOTE: Training &Validation Split has been done in the kNN model.###3.1

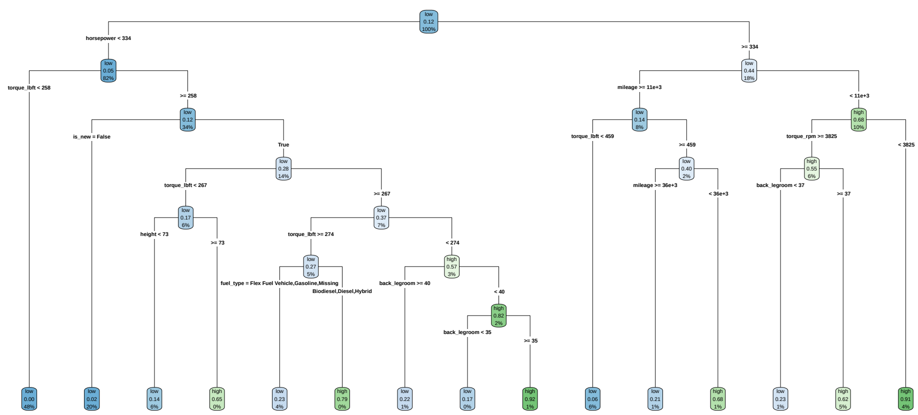
3.1 Classification Trees

```
names(train_df)
```

```
 [1] "back_legroom"        "daysonmarket"        "engine_displacement"
 [4] "frame_damaged"       "franchise_dealer"    "fuel_tank_volume"
 [7] "fuel_type"           "has_accidents"       "height"
[10] "horsepower"          "is_cpo"              "is_new"
[13] "is_oemcpo"           "length"              "maximum_seating"
[16] "mileage"             "owner_count"         "wheelbase"
[19] "width"               "power_rpm"           "torque_lbft"
[22] "torque_rpm"          "price_nom"
```

```r
class_tr <- rpart(price_nom ~ .,
                  data = train_df, method = "class",
                  maxdepth = 30)
```

```r
rpart.plot(class_tr, type=4)
```



3.2 Model Evaluation

3.2.1 Predict the training set

```r
class_tr_train_predict <- predict(class_tr, train_df,
                                  type = "class")

t(t(head(class_tr_train_predict,10)))
```

```
      [,1]
16383 low
17027 high
933   low
15028 low
```

```
21732 low
20324 low
13979 low
6987  low
8247  low
9034  high
Levels: low high
```

```r
confusionMatrix(class_tr_train_predict, train_df$price_nom, positive = "high")
```

```
Confusion Matrix and Statistics

          Reference
Prediction   low  high
      low  13089   518
      high   446  1358

               Accuracy : 0.9374
                 95% CI : (0.9335, 0.9412)
    No Information Rate : 0.8783
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.7025

 Mcnemar's Test P-Value : 0.02221

            Sensitivity : 0.72388
            Specificity : 0.96705
         Pos Pred Value : 0.75277
         Neg Pred Value : 0.96193
             Prevalence : 0.12173
         Detection Rate : 0.08812
   Detection Prevalence : 0.11706
      Balanced Accuracy : 0.84546

       'Positive' Class : high
```

```r
levels(train_df$price_nom)
```

```
[1] "low"  "high"
```

3.2.2 Predict the validation set

```
class_tr_valid_predict <- predict(class_tr, valid_df,
                                  type = "class")
t(t(head(class_tr_valid_predict,10)))
```

```
   [,1]
15 low
18 low
20 low
25 high
30 high
33 low
36 low
38 low
41 low
43 low
Levels: low high
```

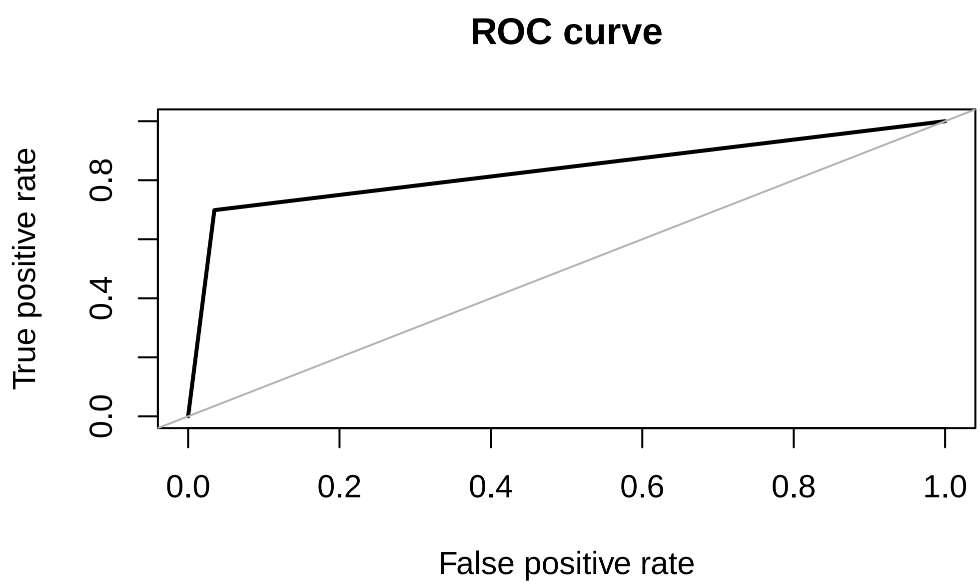
```
confusionMatrix(class_tr_valid_predict, valid_df$price_nom,positive = "high")
```

```
Confusion Matrix and Statistics

          Reference
Prediction  low high
      low  5621  236
      high  202  547

               Accuracy : 0.9337
                 95% CI : (0.9274, 0.9396)
    No Information Rate : 0.8815
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6766

 Mcnemar's Test P-Value : 0.1148

            Sensitivity : 0.6986
            Specificity : 0.9653
         Pos Pred Value : 0.7303
```

```
         Neg Pred Value : 0.9597
             Prevalence : 0.1185
         Detection Rate : 0.0828
   Detection Prevalence : 0.1134
      Balanced Accuracy : 0.8320

       'Positive' Class : high
```

3.3 Model Evaluation

```
ROSE::roc.curve(valid_df$price_nom, class_tr_valid_predict)
```




```
Area under the curve (AUC): 0.832
```

3.3.1 Weighted sampling

```
str(train_df)
```

```
'data.frame':   15411 obs. of  23 variables:
 $ back_legroom       : num  39.5 38.3 40.3 37.6 39.4 38.6 39.8 35.1 35.2 43.6 ...
 $ daysonmarket       : int  38 156 10 39 37 10 48 169 54 18 ...
 $ engine_displacement: int  3600 3600 3600 3600 3300 3600 3500 2400 5300 3500 ...
 $ frame_damaged      : Factor w/ 3 levels "False","Missing",..: 1 1 1 1 1 2 1 1 2 2 ...
 $ franchise_dealer   : Factor w/ 2 levels "False","True": 1 2 2 2 2 2 1 2 2 2 ...
 $ fuel_tank_volume   : num  19.4 22 26 15.8 18.8 24.6 18.6 12.7 24 26 ...
 $ fuel_type          : Factor w/ 6 levels "Biodiesel","Diesel",..: 4 4 3 6 4 4 4 4 4 4 ...
 $ has_accidents      : Factor w/ 3 levels "False","Missing",..: 1 1 1 1 1 2 1 3 2 2 ...
 $ height             : num  66 75 78.5 58.7 66.5 69.3 71 66.5 75.6 77.2 ...
 $ horsepower         : int  310 285 305 295 290 295 290 180 355 375 ...
 $ is_cpo             : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
 $ is_new             : Factor w/ 2 levels "False","True": 1 2 1 1 1 2 1 1 2 2 ...
 $ is_oemcpo          : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
 $ length             : num  190 218 229 192 187 ...
 $ maximum_seating    : int  5 5 6 5 7 5 7 5 6 6 ...
 $ mileage            : num  22666 85 51410 45279 35600 ...
 $ owner_count        : int  1 0 1 1 1 0 2 1 0 0 ...
 $ wheelbase          : num  112 137 140 108 109 ...
 $ width              : num  75 73.8 79.4 73.6 74.4 84.8 90.2 79.6 81.2 96.8 ...
 $ power_rpm          : int  6700 6400 5600 6350 6400 6400 6500 6400 5600 5750 ...
 $ torque_lbft        : int  271 260 410 262 252 260 255 175 383 400 ...
 $ torque_rpm         : int  4900 4800 3950 4250 5300 4000 4000 3900 4100 4500 ...
 $ price_nom          : Factor w/ 2 levels "low","high": 1 2 1 1 1 1 1 1 1 2 ...
 - attr(*, "na.action")= 'omit' Named int [1:7216] 1 2 3 5 13 16 19 26 31 45 ...
  ..- attr(*, "names")= chr [1:7216] "1" "2" "3" "5" ...
```

```r
# List of columns to convert to factors
columns_to_factorize <- c("frame_damaged", "franchise_dealer", "fuel_type", "has_accidents
# Convert all listed columns to factors
train_df[columns_to_factorize] <- lapply(train_df[columns_to_factorize], as.factor)
```

```r
# List of columns to convert to factors
columns_to_factorize <- c("frame_damaged", "franchise_dealer", "fuel_type", "has_accidents

# Convert all listed columns to factors in valid_df
valid_df[columns_to_factorize] <- lapply(valid_df[columns_to_factorize], as.factor)
```

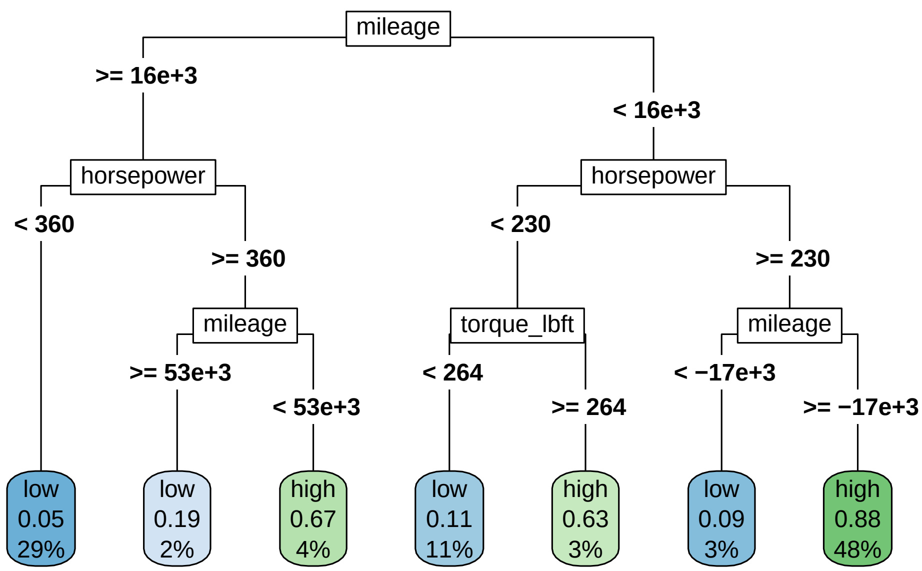
```r
train_df_rose <- ROSE(price_nom ~ .,
                      data = train_df, seed = 666)$data
```

```r
table(train_df_rose$price_nom)
```

```
 low high
7629 7782
```

3.3.1 Weighted data decision tree

```r
class_tr_2 <- rpart(price_nom ~ .,
                    data = train_df_rose, method = "class",
                    maxdepth = 10)

rpart.plot(class_tr_2, type = 5)
```



Predict training set

```r
class_tr_2_train_predict <- predict(class_tr_2, train_df_rose,
                                    type = "class")

summary(class_tr_2_train_predict)
```

```
 low high
6803 8608
```

```r
class_tr_2_train_predict <- as.factor(class_tr_2_train_predict)
train_df_rose$price_nom <- as.factor(train_df_rose$price_nom)
confusionMatrix(class_tr_2_train_predict, train_df_rose$price_nom, positive = "high")
```

```
Confusion Matrix and Statistics

          Reference
Prediction  low high
      low  6299  504
      high 1330 7278

               Accuracy : 0.881
                 95% CI : (0.8758, 0.8861)
    No Information Rate : 0.505
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7617

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9352
            Specificity : 0.8257
         Pos Pred Value : 0.8455
         Neg Pred Value : 0.9259
             Prevalence : 0.5050
         Detection Rate : 0.4723
   Detection Prevalence : 0.5586
      Balanced Accuracy : 0.8805

       'Positive' Class : high
```

Predict Validation set

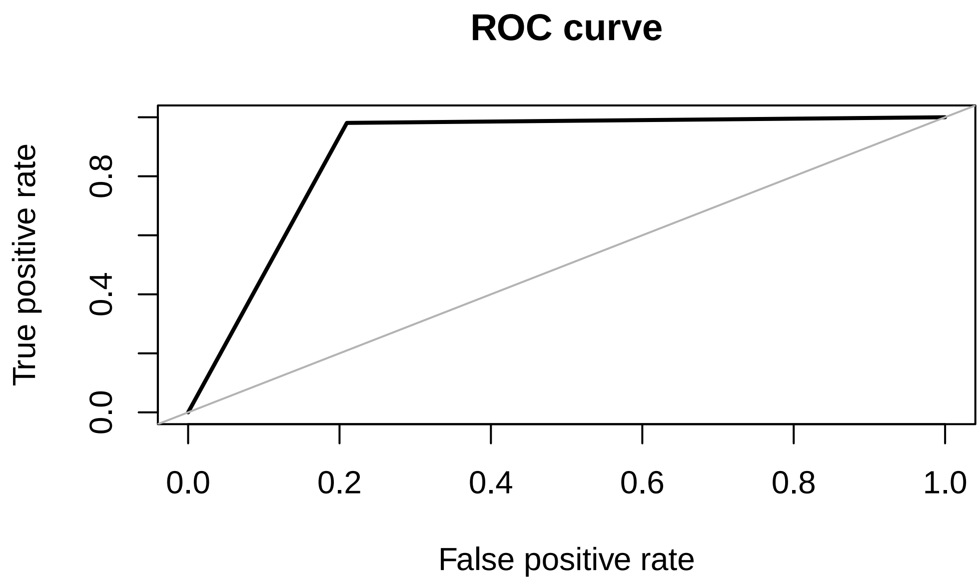
```r
class_tr_2_valid_predict <- predict(class_tr_2, valid_df,
                                    type = "class")

summary(class_tr_2_valid_predict)
```

```
 low high
4617 1989
```

```r
class_tr_2_valid_predict <- as.factor(class_tr_2_valid_predict)
valid_df$price_nom <- as.factor(valid_df$price_nom)
confusionMatrix(class_tr_2_valid_predict, valid_df$price_nom, positive = "high")
```

```
Confusion Matrix and Statistics

          Reference
Prediction  low high
      low  4602   15
      high 1221  768

               Accuracy : 0.8129
                 95% CI : (0.8033, 0.8222)
    No Information Rate : 0.8815
    P-Value [Acc > NIR] : 1

                  Kappa : 0.4627

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9808
            Specificity : 0.7903
         Pos Pred Value : 0.3861
         Neg Pred Value : 0.9968
             Prevalence : 0.1185
         Detection Rate : 0.1163
   Detection Prevalence : 0.3011
      Balanced Accuracy : 0.8856

       'Positive' Class : high
```

```r
ROSE::roc.curve(valid_df$price_nom, class_tr_2_valid_predict)
```




```
Area under the curve (AUC): 0.886
```

4. The new prediction

The weighted data decision tree has a higher roc/ higher sensitivy for both the training &val-idation sets,

```
cars2 <- read.csv('car_test_5.csv', header=TRUE)
```

```
cars2$frame_damaged <- factor(cars2$frame_damaged, levels = levels(train_df_rose$frame_dam
cars2$has_accidents <- factor(cars2$has_accidents, levels = levels(train_df$has_accidents)
```

```
class2_predict <- predict(class_tr_2, cars2, type = "class")
class2_predict
```

```
   1    2    3    4    5    6
high  low  low high  low high
Levels: low high
```

```r
car2_predict <- predict(class_tr_2, newdata = cars2, type = "prob")
car2_predict
```

```
        low       high
1 0.1247987 0.87520129
2 0.9476757 0.05232428
3 0.9476757 0.05232428
4 0.1247987 0.87520129
5 0.9476757 0.05232428
6 0.1247987 0.87520129
```

5. Explantion

Brief Problem Description:

Lightning McQueen and Mator have the used car business in Radiator Springs, but they're missing one key detail: exact prices for their cars. Instead, they've got a broad 'high' or 'low' price label to work with. They're counting on us to predict these labels accurately, as this will help them market the cars correctly.

Objective:

The aim is to create a predictive model that can classify used cars into 'high' or 'low' price categories based on the provided data set. Parts of the data will be used in a training and validation process for two models, and the chosen model will play an essential part in predicting the price categories for a new set of customer data sets, which will have a direct impact on the used car marketing and sales strategy of our client.

Describe the Data:

First, we cleaned the data before making predictions. We checked for gaps and classified the data to help our models understand the difference between a 'high' and a 'low' priced car.

Our extensive dataset includes 30,000 detailed records of used cars, each of which is distinguished by 60 variables. We've narrowed these down to the 23 most influential variables that influence a car's perceived value and, thus, its price category. This includes metrics such as mileage, historical data such as previous accidents, and quality factors such as certification status. We also consider variables that influence customer demand, such as the type of fuel the vehicle use, how long it's been on the market, and whether it's associated with a dealer.

These variables we chose are not just random data points; they are factors which can influence a buyer's decision, influencing the price category in the market.

The models: kNN and Decision Trees

First, we set missing values as "missing", important for the k-Nearest Neighbors (kNN) model, which demands a complete dataset. We converted categorical variables into different levels for model interpretation. We split our dataset into training and validation parts to transform our target price variable into 'low' and 'high' categories. This split allows for model training and performance evaluation on separate datasets to ensure good prediction for new customer datasets.

The cleaned dataset is now prepared for the next step: developing and assessing the performance of two predictive models - kNN and Decision Trees.

kNN Models

We selected the kNN because of its effectiveness in classification tasks. It's suitable for datasets with a mix of variable types like ours and classifies based on similarity measures. In the model, we normalized numerical variables to balance their influence and converted categorical variables into factors for kNN. We set k=5 to balance detail capture and overfitting prevention. We also trained our kNN model, tested its accuracy with a confusion matrix, and used a fixed random seed for consistency. The training results appeared intriguing, indicating that our model can effectively differentiate between 'high' and 'low' price categories. A confusion matrix and ROC curves were used to assess the performance of our KNN model. The Accuracy, Specificity, Sensitivity, and ROC curves will be examined for results.

	Accuracy	*Specificity*	*Sensitivity*
Training Set	95.78%	97.976%	79.904%
Valid Set	94.7%	97.48%	74.07%

Accuracy: The model is very good at classifying cars correctly, as shown by its high accuracy rates of 95.78% on the training set and 94.7% on the validation set. Its high accuracy rates show that the model can effectively apply what it has learned from the training data to new, unseen data.

Specificity: Both sets of data show that the model has high specificity (97.976% for training and 97.48% for validation), which means it is very good at finding cars that should be labeled as “low” price. In practical terms, this means the model is not quite good at classifying a car as ‘high’ price unless the data strongly supports it.

Sensitivity: The model’s sensitivity is 79.904% for training and 74.07% for validation, which is lower than its specificity but still good. This means that the model is pretty good at finding “low” price cars, but not so good at finding “high” price cars. It is slightly less consistent with ‘high’ price cars, which may affect the business if ‘high’ price cars are misclassified and undervalued.

ROC AUC : An AUC value of 0.858 means that the model does a great job of telling the difference between “low” and “high” price ranges. AUC values that are higher would mean that the ability to tell the difference is even better.

In summary, the kNN model is a dependable method for predicting the price category of cars, yet it shows somewhat reduced sensitivity. There is a need to enhance the model’s ability to correctly identify ‘high’ price cars. This step is important for maximizing profits in the car resale business, as missing out on high-value sales opportunities could result in significant revenue loss, in terms of business.

Decision Tree Model

Next is our DTM, and a classification tree is trained with **maxdepth = 30**. The model is evaluated on both training and validation data sets with confusion matrices and ROC curves.

Results

	Accuracy	*Specificity*	*Sensitivity*
Training Set	93.74%	96.71%,	72.39%
Valid Set	93.37%,	96.53%	69.86%

Accuracy: The model achieved 93.74% on the training set and 93.37% on the validation set, showing a strong fit to the data.

Specificity: High specificity on both sets (above 96%) shows the model’s precision in identifying ‘low’ price cars.

Sensitivity: Sensitivity was lower, suggesting it was less adept at catching ‘high’ price cars.

Weighted model results

Based on these results, we decided to use weighted sampling due to its low sensitivity and imbalance which may cause biases later on when we continue with our predictions. Using weighted sampling to address the model's bias towards the 'low' price majority class has improved its performance. The adjustment improved the model's sensitivity, increasing its ability to accurately identify 'high' price cars. This improvement is crucial for the business to prevent revenue loss from undervalued sales.

	Accuracy	*Specificity*	*Sensitivity*
Training Set	88.1%	82.57%,	93.52%
Valid Set	81.29%,	79.03%	98.08%

As the table shown above, with sensitivity scores rising to 93.52% in the training set and 98.08% in the validation set, the model has become more valuable for our client's business. This means that the weighted model is particularly adept at identifying cars with 'high' prices. The AUC value also supports the strong discriminative ability of the weighted model.

The chosen model

Lastly, we decided to choose Decision Trees over kNN for several reasons which include its effectives and the comparisons:

1. Precision in Pricing: The kNN model's sensitivity rate on the validation set was 74.07%, while the decision tree model achieved a very high 98.08%. This high level of precision is crucial for correctly identifying high-value cars, which prevents potential losses from under pricing, thereby ensuring maximum revenue from our sales.
2. The insights from the decision tree model align with our client business's approach to differentiate and market cars in 'high' and 'low' price categories. This combination between analysis and strategy enables more targeted marketing efforts and more effective pricing strategies that resonate with market demands.
3. The AUC metric of the model confirms its strong discriminating power in distinguishing between the price categories. In a business where differences can have a significant financial impact, the model's great classification capability ensures that pricing decisions are reliable.

Given these reasons, the weighted decision tree model is not only a statistical tool but also a part of our business strategy, allowing us to make data-driven decisions that will enhance profitability and secure our position in the competitive used car market.

Predictions for the new customers using your best model

Our decision tree model has analyzed the new customer data set and provided us with great predictions that will guide our pricing strategy. It has identified cars 1, 4, and 6 as belonging to the 'high' price category. This classification comes with a high probability of about 87.52%, giving us a strong assurance that these vehicles possess the qualities that obtain a premium in the market.

Conversely, cars 2, 3, and 5 are predicted to be 'low' price, with the model assigning a high certainty of approximately 94.77% to these predictions. This level of confidence from the model suggests that these cars, are better positioned for a high price segment of our customer base.

With this information at our service, we can make sure that each car is sold to the right customer at the right price by optimizing the prices of our inventory. These predictions allow us to fine-tune our sales approach, maximizing profitability while meeting the diverse needs of our buyers.