Classification

Jovanni Ochoa

February 10 2023

Load the data

#rm(list = ls(all.names = TRUE)) here we clean the data as we load it in taking out na's and dividing into training and testing. THen making the naive process

```
df <- read.csv(file = 'desktop/autos.csv', header=TRUE, stringsAsFactors = TRUE)
df <- df[,c(12, 13, 15, 16)]
df$model <- factor(df$model)
df$kilometer <- factor(df$kilometer)
df$fuelType <- factor(df$fuelType)
df$brand <- factor(df$brand)

#take out NA's
df <- df[!is.na(df$model),]
df <- df[!is.na(df$kilometer),]
df <- df[!is.na(df$fuelType),]
df <- df[!is.na(df$brand),]

#Divide into 80/20 train/test
set.seed(1234)
i <- sample(1:nrow(df), 0.8*nrow(df), replace=FALSE)
train <- df[i,]
test <- df[-i,]

#Naive Bayes process
library(e1071)
nb1 <- naiveBayes(df[,-2], df[,2], data=train)
pred <- predict(nb1, newdata=test[,-2], type="raw")
# look at first 5 (actual: 0 1 1 1 0)
pred[1:5,]
```

```
##             5000       10000       20000       30000       40000      50000
## [1,] 0.004381390 0.008875470 0.041677408 0.024843330 0.054999600 0.05382230
## [2,] 0.009436185 0.001206743 0.003218491 0.002708646 0.003596737 0.00521000
## [3,] 0.005440858 0.002044394 0.008866069 0.010010349 0.011684568 0.01891356
## [4,] 0.011162961 0.003417099 0.013180583 0.015980114 0.015737632 0.01819100
## [5,] 0.015981003 0.001627089 0.011509849 0.012078059 0.008519299 0.01592655
##            60000       70000      80000      90000     100000     125000
## [1,] 0.054714179 0.043050170 0.07126116 0.08292429 0.04901343 0.12780981
## [2,] 0.007057689 0.009910983 0.01319633 0.01926766 0.02455059 0.06740371
## [3,] 0.031386409 0.027728180 0.02851287 0.05168783 0.05361872 0.18602365
## [4,] 0.023965972 0.022741384 0.02802539 0.03300227 0.04376261 0.10986146
## [5,] 0.016608368 0.029909790 0.01908800 0.04677989 0.05743622 0.19424480
##         150000
## [1,] 0.3826275
## [2,] 0.8332362
## [3,] 0.5640825
## [4,] 0.6609715
## [5,] 0.5702911
```

Calculate probability

```
# predict probability
pred_prob <- predict(nb1, newdata = test, type = "raw")

# calculate likelihood
get_model_likelihood <- function(model, kilometer, fuelType, brand) {
  new_data <- data.frame(model = model, kilometer = kilometer, fuelType = fuelType, bran
d = brand)
  prob <- predict(nb1, newdata = new_data, type = "raw")
  names(prob) <- levels(nb1$apriori)
  return(prob)
}

# how to get likelyhood or any ligical query
get_model_likelihood("100", "150,000", "benzin", "audi")
```

```
##             5000       10000     20000     30000       40000       50000
## [1,] 0.003985234 0.005667286 0.0186623 0.0154534 0.004513151 0.006338559
##            60000       70000     80000       90000     100000     125000
## [1,] 0.004294101 0.004252793 0.0131492 0.005801449 0.01634933 0.03135177
##         150000
## [1,] 0.8701814
```

These predictions are interestingly low. This is evident by looking at the accumulation of the data. I think this is because there is such a large amount of data that it's somewhat messing up the process.

Apply to the first 5 test observations

```
summary(train)
```

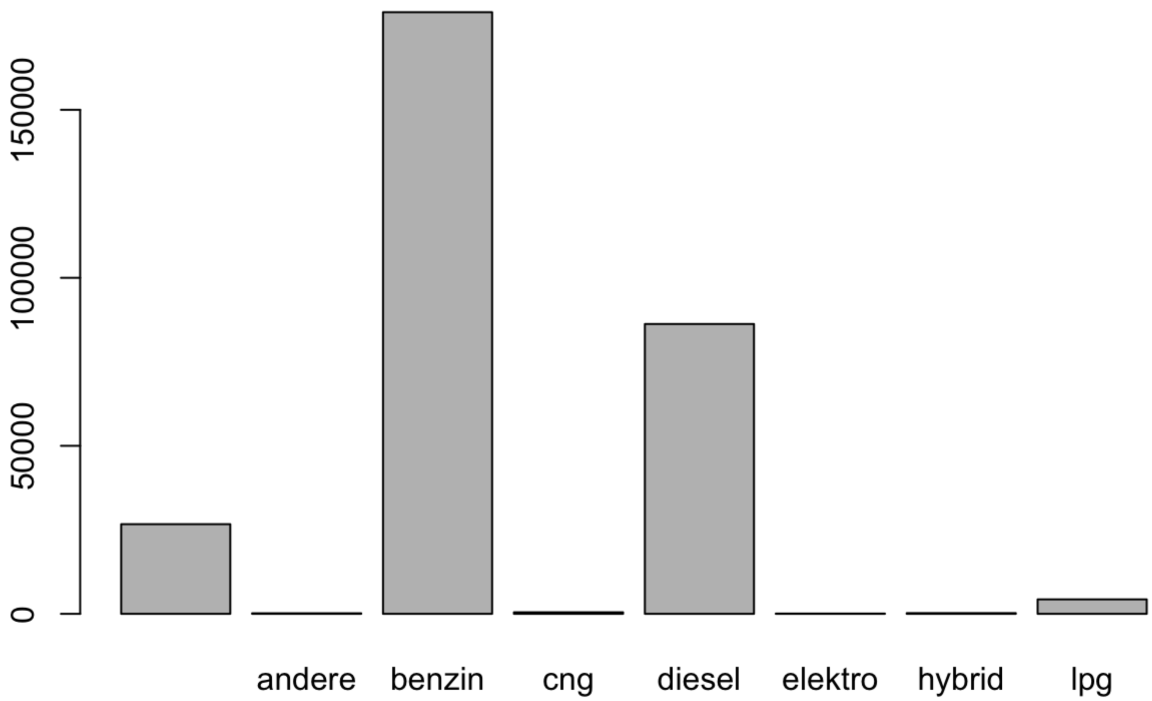
```
##      model          kilometer         fuelType              brand
##   golf    : 24179   150000 :192474   benzin :179054   volkswagen   :63794
##   andere : 21065   125000 : 30367   diesel : 86254   bmw          :32337
##   3er    : 16487   100000 : 12808          : 26681   opel         :32082
##          : 16193   90000  : 10093   lpg    :  4305   mercedes_benz:28245
##   polo   : 10484   80000  :  8907   cng    :   463   audi         :26193
##   corsa  :  9995   70000  :  7816   hybrid :   217   ford         :20526
##   (Other):198819   (Other): 34757   (Other):   248   (Other)      :94045
```

```
head(train)
```

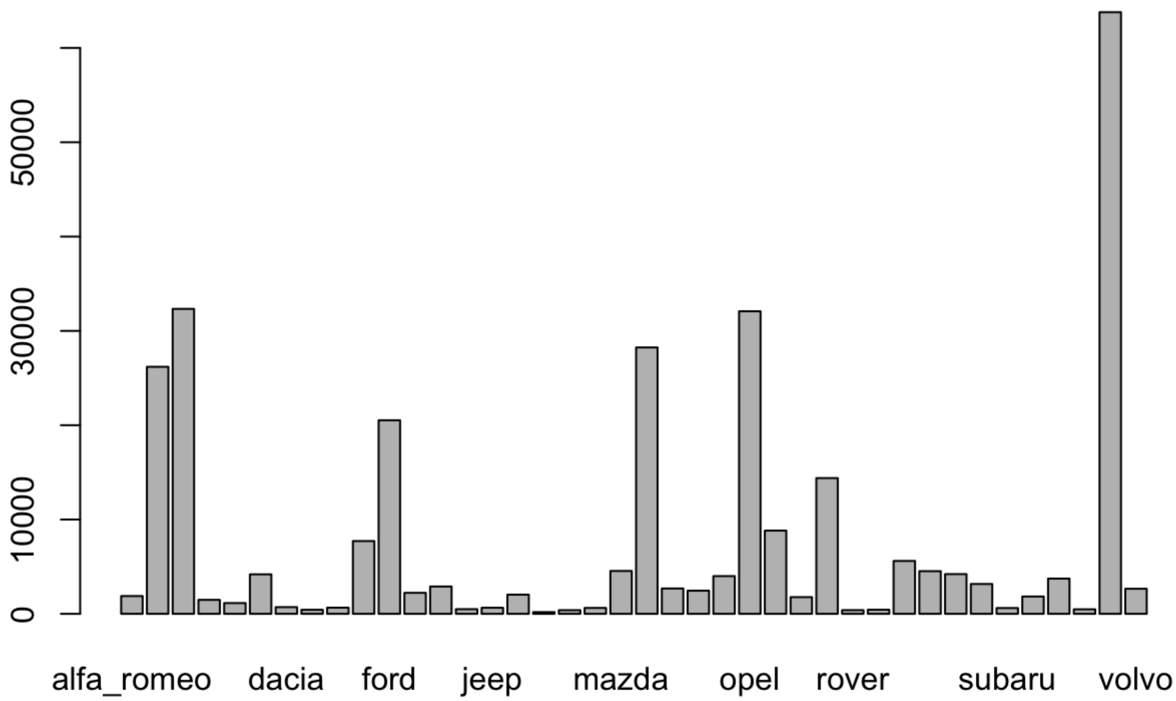
	model <fct>	kilometer <fct>	fuelType <fct>	brand <fct>
237392	2_reihe	150000	benzin	peugeot
106390	3er	150000	benzin	bmw
304108	polo	30000	benzin	volkswagen
295846	golf	80000	benzin	volkswagen
126055	altea	125000	benzin	seat
345167	a4	150000	benzin	audi

6 rows

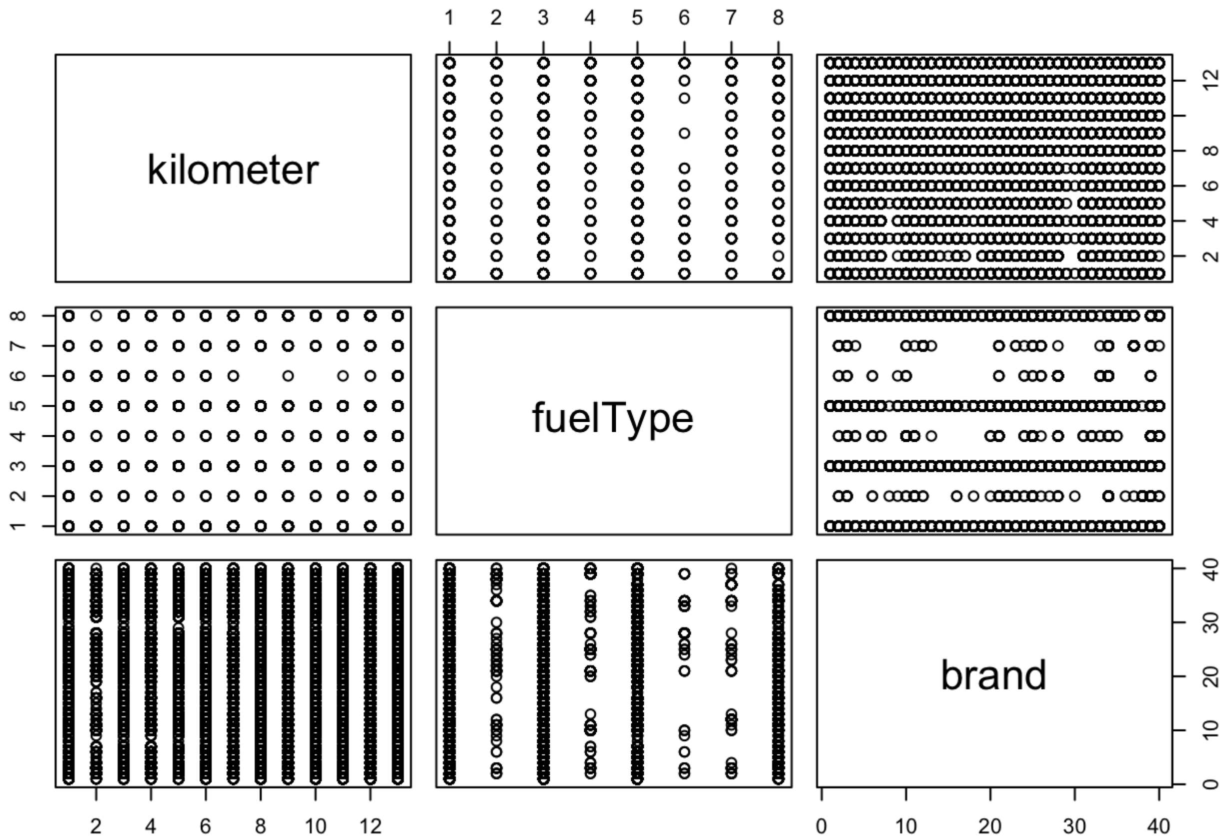
```
plot(train$fuelType)
```




```
barplot(table(train$brand))
```


50000
30000
10000
0
alfa_romeo
dacia
ford
jeep
mazda
opel
rover
subaru
volvo

```
plot(train[, -1])
```



Let's look at just the first 5 test observations.

```
# regression model
model <- glm(model ~ ., data = train, family = "binomial")
summary(model)
```

```
##
## Call:
## glm(formula = model ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1408   0.2131   0.2425   0.2751   1.4067
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)           0.52921    0.11052   4.789 1.68e-06 ***
## kilometer10000        1.04039    0.14914   6.976 3.04e-12 ***
## kilometer20000        0.80961    0.08775   9.226  < 2e-16 ***
## kilometer30000        1.45175    0.11106  13.071  < 2e-16 ***
## kilometer40000        1.44876    0.10930  13.255  < 2e-16 ***
## kilometer50000        1.39396    0.09784  14.247  < 2e-16 ***
## kilometer60000        1.45643    0.09422  15.457  < 2e-16 ***
## kilometer70000        1.37387    0.08741  15.717  < 2e-16 ***
## kilometer80000        1.38206    0.08304  16.642  < 2e-16 ***
## kilometer90000        1.29101    0.07743  16.674  < 2e-16 ***
## kilometer100000       1.08372    0.06583  16.463  < 2e-16 ***
## kilometer125000       1.06491    0.05353  19.893  < 2e-16 ***
## kilometer150000       0.67676    0.04356  15.536  < 2e-16 ***
## fuelTypeandere        0.54314    0.27411   1.981 0.047539 *
## fuelTypebenzin        1.78230    0.02118  84.138  < 2e-16 ***
## fuelTypecng           2.02072    0.29353   6.884 5.82e-12 ***
## fuelTypediesel        2.21585    0.02849  77.779  < 2e-16 ***
## fuelTypeelektro       1.90426    0.72387   2.631 0.008521 **
## fuelTypehybrid        2.39447    0.58513   4.092 4.27e-05 ***
## fuelTypelpg           1.68233    0.07898  21.300  < 2e-16 ***
## brandaudi             0.25002    0.10781   2.319 0.020385 *
## brandbmw              0.08996    0.10617   0.847 0.396791
## brandchevrolet       -0.21762    0.15399  -1.413 0.157607
## brandchrysler        -0.08392    0.16550  -0.507 0.612129
## brandcitroen          0.12260    0.12826   0.956 0.339142
## branddacia            0.35660    0.25243   1.413 0.157762
## branddaewoo          -0.38097    0.20898  -1.823 0.068302 .
## branddaihatsu         0.12545    0.20813   0.603 0.546655
## brandfiat             0.13071    0.11616   1.125 0.260490
## brandford             0.34605    0.10903   3.174 0.001505 **
## brandhonda            0.11924    0.14122   0.844 0.398468
## brandhyundai         -0.39107    0.12993  -3.010 0.002613 **
## brandjaguar           0.80599    0.31424   2.565 0.010321 *
## brandjeep             0.55625    0.25917   2.146 0.031849 *
## brandkia             -0.11439    0.14813  -0.772 0.439999
## brandlada            -0.41954    0.31655  -1.325 0.185063
## brandlancia           0.22251    0.26238   0.848 0.396419
## brandland_rover       1.06995    0.35587   3.007 0.002642 **
## brandmazda           -0.04359    0.12209  -0.357 0.721052
## brandmercedes_benz    0.35171    0.10789   3.260 0.001114 **
## brandmini             0.79530    0.17746   4.482 7.41e-06 ***
## brandmitsubishi       0.12290    0.13915   0.883 0.377094
```

```
## brandnissan           0.28177    0.13147   2.143 0.032098 *
## brandopel              0.30755    0.10637   2.891 0.003836 **
## brandpeugeot          -0.46767    0.11057  -4.229 2.34e-05 ***
## brandporsche           0.84558    0.18850   4.486 7.27e-06 ***
## brandrenault           0.39022    0.11133   3.505 0.000456 ***
## brandrover            -1.05379    0.17813  -5.916 3.30e-09 ***
## brandsaab              0.62802    0.30309   2.072 0.038260 *
## brandseat              0.53106    0.12851   4.133 3.59e-05 ***
## brandskoda             0.63962    0.14291   4.476 7.62e-06 ***
## brandsmart            -0.15529    0.12625  -1.230 0.218679
## brandsonstige_autos -19.55802   40.23093  -0.486 0.626865
## brandsubaru            0.07069    0.20685   0.342 0.732551
## brandsuzuki           -0.13129    0.14624  -0.898 0.369297
## brandtoyota            0.30823    0.13773   2.238 0.025226 *
## brandtrabant          -0.48811    0.17115  -2.852 0.004345 **
## brandvolkswagen        0.26737    0.10469   2.554 0.010650 *
## brandvolvo             0.54616    0.15141   3.607 0.000310 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 125727  on 297221  degrees of freedom
## Residual deviance:  96300  on 297163  degrees of freedom
## AIC: 96418
##
## Number of Fisher Scoring iterations: 15
```

```
# test
pred <- predict(model, newdata = test, type = "response")

#check predictions
threshold <- 0.5
pred_labels <- ifelse(pred >= threshold, 1, 0)
actual_labels <- test$model
accuracy <- mean(pred_labels == actual_labels)
print(paste("Accuracy on test set:", round(accuracy, 4)))
```

```
## [1] "Accuracy on test set: 0"
```

It makes sense the model looks like this given what we are tying to look for, but the likelyhood as evident from the standard error is perfect to make predictions from some of the data. I think it’s really good because it has a lot of data to work with, as well as, the data is linear. Some other data however, begs to differ as it has no correlation thus reducing things overall.

###Summary With ths standard error being so low, we can accurately say that the data is significant in many factors. This makes sense because a car will have the same fuel type and brand. The results differ a considerable amount when regards to accuracy. This is undeniably the fact because of the amount of data being gave.

Both an advantage and a disadvantage of Naive is that it considerest hings to be independent of each other. This is most likely not the case. Especially not with the data that I gave. For this, the logistic regression is a lot better. There is also the problem with the amount of data being fed. logistic regression handles it better, while it looks like Naive struggles with large data.

The accuracy is terrible. I think it's because one of some weird inputs being done incorrectly, but I'm not exactly sure. AIC is also too high for this type of data. Usually the lower the better.