Capstone 2: Car-Evaluation

M. Akhtar

Feb 26, 2022

Overview

About this project

This machine learning project is about creating a car evaluation system using the car data from the archive of University of California, Irvine. The goal is to develop a machine learning model that can predict the class of the car based on the features of the car with accuracy as high as possible. Using the randomForest model my recommendation algorithm was able to predict the class from the unknown validation dataset with an accuracy of 0.9595.

About Car-Evaluation DataSet

The car evaluation dataset contains 1727 instances with one outcome and six features:

Outcome

- class: represents unacc, acc, good, vgood.

Features

- buying: represents vhigh, high, med, low.
- maint: vhigh, high, med, low.
- doors: 2, 3, 4, 5more.
- persons: 2, 4, more.
- lug_boot: small, med, big.
- safety: low, med, high.

Method

To develop the recommender system I performed the following tasks:

- Installing the required packages.
- Download the car dataset.
- Organize the car dataset.
- Explore the car dataset.
- Split the car dataset into edx (80%) and validation (20%).
- Use the edx set to develop the algorithms and then use the validation to determine the accuracy to check the performance.
- Build the prediction model including The Generalized Linear Model(glm),KNN model without caret package, KNN Model with Caret package, and RandomForest model.
- Select the best model with greatest accuracy.

1. Installing packages

The following packages were installed.

```
if(!require(readr)) install.packages("readr", repos = "http://cran.us.r-project.org")
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(xfun)) install.packages("xfun", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(rshape2)) install.packages("reshape2", repos = "http://cran.us.r-project.org")
```

```
## package 'reshape2' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Instructor\AppData\Local\Temp\RtmpK4HaR0\downloaded_packages
```

```
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(e1071)) install.packages("e1071", repos = "http://cran.us.r-project.org")
if(!require(digest)) install.packages("digest", repos = "http://cran.us.r-project.org")
if(!require(pander)) install.packages("pander", repos = "http://cran.us.r-project.org")
if(!require(kableExtra)) install.packages("kabelExtra", repos = "http://cran.us.r-project.org")
```

Downloading the data

I downloaded the dataset from: https://archive.ics.uci.edu/ml/machine-learning-databases/car and saved on my machine as cardl.RData. Then I used the following code to upload the car dataset.

```
## Upload data
load("cardl.RData")
```

3. Exploration of data

I checked the cardl data-set if it is clean. Glimpse of cardl data-set provides the following information:

- cardl is a dataframe/datatable with seven columns (variables) and 1727 rows.
- cardl's first six columns are features of the car.
- cardl's Seventh column is class outcome.
- cardl data-set has no missing values.

```
library(tidyverse)

cardl %>% summarize(NA_buying=sum(is.na(buying)),
                    NA_maint=sum(is.na(maint)),
                    NA_doors=sum(is.na(doors)),
                    NA_persons=sum(is.na(persons)),
                    NA_lug_boot=sum(is.na(lug_boot)),
                    NA_safety=sum(is.na(safety)),
                    NA_class=sum(is.na(class)))
```

```
## # A tibble: 1 x 7
##   NA_buying NA_maint NA_doors NA_persons NA_lug_boot NA_safety NA_class
##       <int>    <int>    <int>      <int>       <int>     <int>    <int>
## 1         0        0        0          0           0         0        0
```

```
glimpse(cardl)
```

```
## Rows: 1,727
## Columns: 7
## $ buying   <chr> "vhigh", "vhigh", "vhigh", "vhigh", "vhigh", "vhigh", "vhigh"~
## $ maint    <chr> "vhigh", "vhigh", "vhigh", "vhigh", "vhigh", "vhigh", "vhigh"~
## $ doors    <chr> "2", "2", "2", "2", "2", "2", "2", "2", "2", "2", "2", "2", "~
## $ persons  <chr> "2", "2", "2", "2", "2", "2", "2", "2", "4", "4", "4", "4", "~
## $ lug_boot <chr> "small", "small", "med", "med", "med", "big", "big", "big", "~
## $ safety   <chr> "med", "high", "low", "med", "high", "low", "med", "high", "l~
## $ class    <chr> "unacc", "unacc", "unacc", "unacc", "unacc", "unacc", "unacc"~
```

4. Data preparation

Data preparation involves the following steps:

- Make dl.copy of cardl data.

- Replace “5more” with 5 in doors column 3.
- Replace “more” with 5 in persons column 4.
- Convert categorical dataset of dl.copy into numeric data frame and place in card.dat.

```
library(tidyverse)
dl.copy<-cardl
dl.copy<-dl.copy %>%
  mutate(doors =case_when(doors =="5more" ~ "5",
           TRUE ~ as.character(doors)), doors =                          as.integer(doors))%>%
  mutate(persons = case_when(persons =="more" ~ "5",
           TRUE ~ as.character(persons)),persons =                       as.integer(persons))

car.mat<-data.matrix(dl.copy)
car.dat<-data.frame(car.mat)
glimpse(car.dat)
```

```
## Rows: 1,727
## Columns: 7
## $ buying   <int> 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4~
## $ maint    <int> 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4~
## $ doors    <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ persons  <int> 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5~
## $ lug_boot <int> 3, 3, 2, 2, 2, 1, 1, 1, 3, 3, 3, 2, 2, 2, 1, 1, 1, 3, 3, 3, 2~
## $ safety   <int> 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2~
## $ class    <int> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3~
```

5. Creation of heatmap

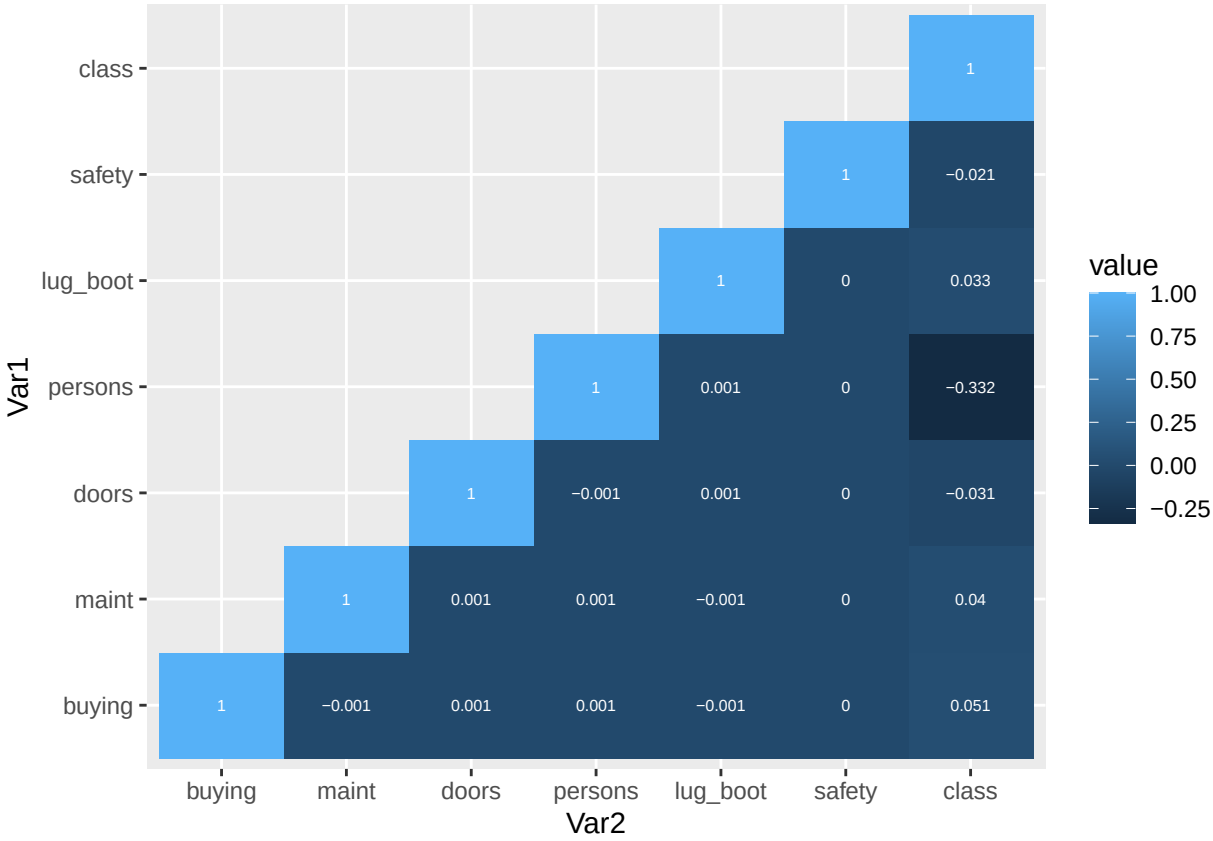
I created a correlation matrix and its corresponding heat-map to check for any evidence of dependence between any two variables.

```
library(reshape2)
x<-round(cor(car.mat),3)
x
```

```
##          buying  maint  doors persons lug_boot safety  class
## buying    1.000 -0.001  0.001   0.001   -0.001  0.000  0.051
## maint    -0.001  1.000  0.001   0.001   -0.001  0.000  0.040
## doors     0.001  0.001  1.000  -0.001    0.001  0.000 -0.031
## persons   0.001  0.001 -0.001   1.000    0.001  0.000 -0.332
## lug_boot -0.001 -0.001  0.001   0.001    1.000  0.000  0.033
## safety    0.000  0.000  0.000   0.000    0.000  1.000 -0.021
## class     0.051  0.040 -0.031  -0.332    0.033 -0.021  1.000
```

```
cormat<-x
cormat<-round((cormat),3)
cormat[!(col(cormat)>=row(cormat))]<-NA
melted_cormat<-melt(cormat,na.rm=TRUE)

ggheatmap<-ggplot(melted_cormat, aes(Var2, Var1,
                                     fill = value))+ geom_raster()
ggheatmap + geom_text(aes(Var2, Var1, label = value)
                      , color = "white", size = 2)
```



From the heatmap, it is evident that there is no significant correlation between any two variables. The highest correlation is -0.332 between outcome, class and feature, persons in the car.

6. Partition of car data

dl.copy of car data set is split into edx (80%) and validation (20%)sets:

- edx set is used in model development
- validation set is NOT used anywhere except when testing the final model.

- edx data-set has integers.

- validation data-set has charcaters.

```
library(caret)
set.seed(1,sample.kind="Rounding")#if usig R 3.5 or eralier, use set.seed(1)
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
testIndex<-createDataPartition(y=dl.copy$class, times=1, p=0.2, list=FALSE)
edx<-dl.copy[-testIndex,]
validation<-dl.copy[testIndex,]
```

```
edx<-as.data.frame(data.matrix(edx))
head(edx)
```

```
##   buying maint doors persons lug_boot safety class
## 1      4     4     2       2        3      3     3
## 2      4     4     2       2        3      1     3
## 3      4     4     2       2        2      2     3
## 4      4     4     2       2        2      3     3
## 5      4     4     2       2        2      1     3
## 6      4     4     2       2        1      2     3
```

```
head(validation)
```

```
## # A tibble: 6 x 7
##   buying maint doors persons lug_boot safety class
##   <chr>  <chr> <int>   <int> <chr>    <chr>  <chr>
## 1 vhigh  vhigh     2       4 big      med    unacc
## 2 vhigh  vhigh     3       2 small    high   unacc
## 3 vhigh  vhigh     3       4 big      low    unacc
## 4 vhigh  vhigh     3       5 med      med    unacc
## 5 vhigh  vhigh     3       5 med      high   unacc
## 6 vhigh  vhigh     3       5 big      low    unacc
```

6.1 Partitioning of validation data

validation data was split into 50% vtest_set, and 50% validation_test set:

- vtest_set has integers.
- validation_test has characters.

```
vtestIndex = sort(sample(nrow(validation), nrow(validation)*.5))
vtest_set<-validation[vtestIndex,]
validation_test<-validation[-vtestIndex,]
head(vtest_set)
```

```
## # A tibble: 6 x 7
##   buying maint doors persons lug_boot safety class
##   <chr>  <chr> <int>   <int> <chr>    <chr>  <chr>
## 1 vhigh  vhigh     2       4 big      med    unacc
## 2 vhigh  vhigh     3       2 small    high   unacc
## 3 vhigh  vhigh     3       5 med      med    unacc
## 4 vhigh  vhigh     3       5 med      high   unacc
## 5 vhigh  vhigh     4       5 small    med    unacc
## 6 vhigh  vhigh     5       2 big      low    unacc
```

```
head(validation_test)
```

```
## # A tibble: 6 x 7
##   buying maint doors persons lug_boot safety class
##   <chr>  <chr> <int>   <int> <chr>    <chr>  <chr>
```

```
## 1 vhigh  vhigh     3       4 big      low    unacc
## 2 vhigh  vhigh     3       5 big      low    unacc
## 3 vhigh  vhigh     4       4 small    high   unacc
## 4 vhigh  vhigh     4       4 med      med    unacc
## 5 vhigh  vhigh     4       4 med      high   unacc
## 6 vhigh  vhigh     4       5 small    low    unacc
```

6.2 Conversion of data into Factors

```
library(dplyr)
#edx has integers, convert into factors
train_set<-as.data.frame(edx)%>%
 select(buying,maint,doors,persons,lug_boot,safety,class)%>%
 mutate(buying=factor(buying),
        maint=factor(maint),
        lug_boot=factor(lug_boot),
        safety=factor(safety),
        class=factor(class))
#convert vtest_set into integers, data frame, and then into factors
test_set<-as.data.frame(data.matrix(vtest_set))
test_set<-test_set%>%
 select(buying,maint,doors,persons,lug_boot,safety,class)%>%
 mutate(buying=factor(buying),
        maint=factor(maint),
        lug_boot=factor(lug_boot),
        safety=factor(safety),
        class=factor(class))
#Convert validation_set into factors

validation_set<-as.data.frame(data.matrix(validation_test))%>%
 select(buying,maint,doors,persons,lug_boot,safety,class)%>%
        mutate(buying=factor(buying),
         maint=factor(maint),
        lug_boot=factor(lug_boot),
        safety=factor(safety),
        class=factor(class))
```

7. Modeling techniques

Since the data is categorical, I considered the following classification techniques:

- Multinomial logistic model
- k-NN classification without the caret package
- K-NN classification with caret package
- Random Forest

7.1 Multinomial logistic Model

I used multinom function from the nnet r package to estimate a multinomial logistic regression model, because: - The outcome, class has four categories. - Multinomial logistic regression does not assume normality,

linearity, or homoscedasticity.

```
library(nnet)
library(e1071)
mnm.fit<-multinom(class~.,data=train_set)
```

```
## # weights:  56 (39 variable)
## initial  value 1914.472513
## iter  10 value 863.200014
## iter  20 value 538.124042
## iter  30 value 403.867948
## iter  40 value 330.511228
## iter  50 value 310.588888
## iter  60 value 309.637704
## iter  70 value 309.562005
## iter  80 value 309.546895
## final  value 309.546888
## converged
```

```
y_hat_mnm<-predict(mnm.fit,test_set)
accuracy_mnm<-confusionMatrix(y_hat_mnm,validation_set$class)$overall[["Accuracy"]]
accuracy_mnm
```

```
## [1] 0.6416185
```

```
accmat<-table("pred" = y_hat_mnm, "class" = factor(validation_set$class))
accmat
```

```
##     class
## pred  1  2  3  4
##    1 14  1 21  1
##    2  1  2  2  1
##    3 17  5 95  5
##    4  3  0  5  0
```

7.2 K Nearest Neighbor (KNN) Model

KNN is a non-parametric Supervised Learning Model that uses input features to predict the outcome. It is based on the similarity of the features. Knn algorithm stores k features with nearest neighbors and classifies new cases by majority vote of its k neighbors.KNN works with factors.
Since the data points are in single digit, I did not normalize the data. KNN model that works:

- Without caret package and with specific value of k such as as the square root of number of features in the training sample.
- With caret package and tuning parameters.

```
cardat_copy<-dl.copy
cardat_mat<-data.matrix(cardat_copy)
cardat<-data.frame(cardat_mat)
```

```r
#Remove the response variable "class"
class_outcome<-cardat%>%select(class)
car_features<-cardat%>%select(-class)
set.seed(2021)
sample_size<-floor(0.80*nrow(car_features))
train_ind<-sample(seq_len(nrow(car_features)),size = sample_size)

class_pred_train<-car_features[train_ind,]
class_pred_test<-car_features[-train_ind,]
class_outcome_train<-class_outcome[train_ind,]
class_outcome_test<-class_outcome[-train_ind,]
k<-floor(sqrt(sample_size))
k
```

```
## [1] 37
```

```r
library(class)
class_pred_knn<-knn(train=class_pred_train,test=class_pred_test,cl=class_outcome_train, k)
# Model evaluation
accuracy_without_caret<-confusionMatrix(class_pred_knn,factor(class_outcome_test))$overall["Accuracy"]
accuracy_without_caret
```

```
##  Accuracy
## 0.7572254
```

```r
accmat<-table("pred" = class_pred_knn, "class"
              = factor(class_outcome_test))
accmat
```

```
##     class
## pred   1   2   3   4
##    1  15   5   0   2
##    2   0   0   0   0
##    3  66   6 241   5
##    4   0   0   0   6
```

Knn_Model with caret package to tuning parameter

```r
library(caret)
train_knn<-train(class_pred_train,class_outcome_train, method = "knn",preProcess =c("center","scale"))
train_knn
```

```
## k-Nearest Neighbors
##
## 1381 samples
##    6 predictor
##
## Pre-processing: centered (6), scaled (6)
## Resampling: Bootstrapped (25 reps)
```

```
## Summary of sample sizes: 1381, 1381, 1381, 1381, 1381, 1381, ...
## Resampling results across tuning parameters:
##
##   k  RMSE       Rsquared   MAE
##   5  0.5985835  0.5360706  0.3099331
##   7  0.5951667  0.5391798  0.3336267
##   9  0.5989157  0.5346292  0.3521698
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 7.
```

```
k<-train_knn$bestTune
class_pred_knn<-knn(train=class_pred_train,test=class_pred_test,cl=class_outcome_train, k)
#Model Evaluation
accuracy_with_caret<-confusionMatrix(class_pred_knn,factor(class_outcome_test))$overall["Accuracy"]

accuracy_with_caret
```

```
##  Accuracy
## 0.9075145
```

```
accmat<-table("pred" = class_pred_knn, "class_labels" = factor(class_outcome_test))
accmat
```

```
##     class_labels
## pred   1   2   3   4
##    1  55   4   0   2
##    2   0   7   0   0
##    3  26   0 241   0
##    4   0   0   0  11
```

Random Forrest Model Random Forest is a very popular Machine Learning Model because:

- It provides good efficiency with both numerical and categorical data.
- Algorithm divides the data into sub-samples, and then on each sub-sample fits a decision tree.
- It takes the averaging of outcomes of various decision trees to predict the final outcome.

```
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
library(caret)
library(randomForest)
library(datasets)
set.seed(1971)

sample_size<-floor(0.80*nrow(car_features))
train_ind<-sample(seq_len(nrow(car_features)),size = sample_size)
class_pred_train<-car_features[train_ind,]
class_pred_test<-car_features[-train_ind,]
class_outcome_train<-class_outcome[train_ind,]
class_outcome_test<-class_outcome[-train_ind,]
```

```
rf<-randomForest(factor(class_outcome_train)~.,data=class_pred_train)

#Prediction & Confusion Matrix-test data
rf_class_pred<-predict(rf,class_pred_test)
accuracy_rf<-confusionMatrix(rf_class_pred,factor(class_outcome_test))$overall["Accuracy"]
accuracy_rf
```

```
##  Accuracy
## 0.9595376
```

```
accmat<-table("pred" =  rf_class_pred, "class_labels" = factor(class_outcome_test))
accmat
```

```
##     class_labels
## pred   1   2   3   4
##    1  69   9   0   0
##    2   0   7   0   0
##    3   4   0 241   0
##    4   1   0   0  15
```

8. Results

```
library(kableExtra)
library(data.table)
percent_accuracy<-data.frame(Model=c("(Multinomial logistic Model","KNN Model without caret","KNN Model
  Accuracy=c(accuracy_mnm,accuracy_without_caret,
                 accuracy_with_caret, accuracy_rf))
percent_accuracy
```

```
##                          Model  Accuracy
## 1  (Multinomial logistic Model 0.6416185
## 2      KNN Model without caret 0.7572254
## 3 KNN Model with Caret Package 0.9075145
## 4                 RandomForest 0.9595376
```

9. Conclusion & Future Work

Multinomial logistic model provides the least accuracy of around 64% while the Random Forest Model provides the greatest accuracy of around 96%. I will implement Support Vector Machines(SVMs) in R to see if the accuracy more than 96% can be achieved.