Data Driven Knowledge Discovery Final Exam - Summer 2022

Analysis of Craigslist Vehicle Postings Data

Rahul Golhar

8/6/2022

Contents

Introduction

This is the document for final examination submitted by Rahul Golhar for the course ISTE 780 - Data Driven Knowledge Discovery taught by Prof. Mick McQuaid in the Summer 2022 term at Rochester Institute of Technology.

We start by reading the data into 2 data frames, one for metadata from file 'vehicles.dat' and one for the actual data from 'vehicles.csv'. Each column is also given a data type while reading.

```
# metadata file path
metadataFilename = "./data/vehicles.dat"

# vehicles data file path
vehicledataFilename = "./data/vehicles.csv"

# read metadata table from file
metadata <- data.table::fread(metadataFilename, colClasses = c("integer", "character", "character"))

# read vehicles data table from file
vehicleData <- data.table::fread(vehicledataFilename, header = TRUE,
    colClasses = c(
        "numeric", "character", "character","character",
        "numeric", "numeric", "character", "character",
        "character", "character", "character", "numeric",
        "character", "character", "character","character",
        "character", "character", "character","character",
        "character", "character", "character","numeric",
        "numeric", "character"), na.strings = c("", "NA"))
```

Data Description

The data provided is about craigslist postings of vehicles and each field is described in the table 1. We can observe that there are various fields which give an instinct of being useful for ML models like entry price, entry year, manufacturer of vehicle, model of vehicle, condition of vehicle, number of cylinders, fuel type, miles traveled by vehicle, transmission of vehicle and more. These are some of the major factors the customers usually tend to look at and decide whether to buy the vehicle or not.

```
suppressMessages(library(kableExtra))

# list of names of columns
names <- metadata$V2

# list of description of columns
descriptionOfFields <- metadata$V3

# list of data types of columns
dataTypes <- c(
        "numeric", "character", "character","character",
        "numeric", "numeric", "character", "character",
        "character", "character", "character", "numeric",
        "character", "character", "character","character",
        "character", "character", "character","character",
```

```
        "character", "character", "character","numeric",
        "numeric", "character")

# define the table variable
dataDescTable <- tibble(FIELD=c( names ), DESCRIPTION=c(  descriptionOfFields ), DATATYPE=c(dataTypes))

# display the table
suppressMessages(
kable(dataDescTable,
      booktabs = T,
      caption = "Description of fields in the raw data") %>%
      kable_styling(latex_options = c("HOLD_position")) %>%
      column_spec(1, bold = T, width = "3cm") %>%
      column_spec(2, width = "5 cm") %>%
      column_spec(3, width = "3 cm")
)
```

Table 1: Description of fields in the raw data

FIELD	DESCRIPTION	DATATYPE
id	entry ID	numeric
url	listing URL	character
region	craigslist region	character
region_url	region URL	character
price	entry price	numeric
year	entry year	numeric
manufacturer	manufacturer of vehicle	character
model	model of vehicle	character
condition	condition of vehicle	character
cylinders	number of cylinders	character
fuel	fuel type	character
odometer	miles traveled by vehicle	numeric
title_status	title status of vehicle	character
transmission	transmission of vehicle	character
vin	vehicle identification number	character
drive	type of drive	character
size	size of vehicle	character
type	generic type of vehicle	character
paint_color	color of vehicle	character
image_url	image URL	character
description	listed description of vehicle	character
county	useless column left in by mistake	character
state	state of listing	character
lat	latitude of listing	numeric
long	longitude of listing	numeric
posting_date	date of craigslist listing	character

Data Pre-processing

The data provided needs some cleaning up like removing NaN's, deleting the columns added by mistake and finally having the data ready for applying algorithms.

```
prev_length = nrow(vehicleData)

# remove county column
vehicleData <- subset(vehicleData, select=-county)

# replace nans with unknown
vehicleData$manufacturer <- vehicleData$manufacturer %>% replace_na("unknown")
vehicleData$model <- vehicleData$model %>% replace_na("unknown")
vehicleData$cylinders <- vehicleData$cylinders %>% replace_na("unknown")
vehicleData$title_status <- vehicleData$title_status %>% replace_na("unknown")
vehicleData$drive <- vehicleData$drive %>% replace_na("unknown")
vehicleData$type <- vehicleData$type %>% replace_na("unknown")
vehicleData$state <- vehicleData$state %>% replace_na("unknown")
vehicleData$condition <- vehicleData$condition %>% replace_na("unknown")
vehicleData$paint_color <- vehicleData$paint_color %>% replace_na("unknown")
vehicleData$size <- vehicleData$size %>% replace_na("unknown")


# remove non required columns
remove_cols = c("url", "region_url", "description", "image_url")

vehicleData <- subset(vehicleData,
                      select = -c(url, region_url, description, image_url))

# remove nans
vehicleData <- vehicleData[rowSums(is.na(vehicleData)) == 0,]

# remove duplicate data
vehicleData <- vehicleData[!duplicated(vehicleData), ]

new_length = nrow(vehicleData)
```

As we saw in table 1, there was a column named "county" which we need to remove since it was mistakenly added. The next step is to replace all Nan's with "unknown" in columns: "manufacturer", "model", "cylinders", "title_status", "drive", "type", "state", "condition", "paint_color" and "size". The next task is to drop the NaN's from some columns which aren't used in further Machine Learning purposes. We will only be using a subset of this data for our analysis and the columns that we will be removing are: url, region_url, description, image_url leaving us with the following columns: id, region, price, year, manufacturer, model, condition, cylinders, fuel, odometer, title_status, transmission, VIN, drive, size, type, paint_color, state, lat, long, posting_date. Finally, "de-duplication" operation is performed on the data. So, initially there were 426880 rows and after performing above operations NaN's we now have 256166 i.e. we removed 170714 entries. And we are reamining with following columns: id, region, price, year, manufacturer, model, condition, cylinders, fuel, odometer, title_status, transmission, VIN, drive, size, type, paint_color, state, lat, long, posting_date.

Linear Regression

For this algorithm, we predict "Price" of the car based on various factors using linear regression models. For this task we use columns: "price"(label), "year", "manufacturer", "odometer", "condition", "fuel", "transmission", "drive", "size", "type" and "paint_color". Before we start, we need to get the data ready and factorize the categorical columns i.e. the columns except "year" and "odometer".

```
library(caret)

# get the subset of data
linRegData <- subset(vehicleData,
                     select = c(price, year, manufacturer, odometer,
                                condition, fuel, transmission, drive,
                                size, type, paint_color))

# Factor the data from
linRegData$manufacturer <-linRegData$manufacturer<-as.factor(linRegData$manufacturer)
linRegData$condition <-linRegData$condition<-as.factor(linRegData$condition)
linRegData$fuel <-linRegData$fuel<-as.factor(linRegData$fuel)
linRegData$transmission <-linRegData$transmission<-as.factor(linRegData$transmission)
linRegData$drive <-linRegData$drive<-as.factor(linRegData$drive)
linRegData$size <-linRegData$size<-as.factor(linRegData$size)
linRegData$type <-linRegData$type<-as.factor(linRegData$type)
linRegData$paint_color <-linRegData$paint_color<-as.factor(linRegData$paint_color)
```

Now our data is ready for further steps.

```
set.seed(1)

# get rows to split data for training and testing purposes
train_entries <- sort(sample(nrow(linRegData), nrow(linRegData) * 0.8))

# read data for training and testing sets
train_data = linRegData[train_entries,]
test_data = linRegData[-train_entries,]
```

Once we are done with getting the data ready, we split it into training and testing sets with a split of 80% and 20% respectively. So, now we have a training set of 204932 entries and a testing set of 51234.

Predict Price using entry year and miles travelled by the vehicle

Here, we develop a model to predict "Price" using fields "year" and "odometer" and train it.

```
linreg_model_1 <- lm(formula = price ~
                     year+odometer,
  # year+manufacturer+odometer+condition+fuel+transmission+drive+size+type+paint_color,
                   data = train_data)
```

The summary of this model can be found below:

```
summary(linreg_model_1)
```

```
##
## Call:
## lm(formula = price ~ year + odometer, data = train_data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
##    -30297     -9018     -2969      6519 123433916
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.492e+06  2.256e+05  -6.615 3.72e-11 ***
## year         7.531e+02  1.119e+02   6.730 1.70e-11 ***
## odometer    -3.354e-02  8.146e-03  -4.118 3.83e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 278900 on 204929 degrees of freedom
## Multiple R-squared:  0.0004903,  Adjusted R-squared:  0.0004805
## F-statistic: 50.26 on 2 and 204929 DF,  p-value: < 2.2e-16
```

Let's have a look at the performance of this model:

We calculate Residual Sum of Squares for training data which measures the amount of error remaining between the regression function and the data set after the model has been run.

```
# calculate Residual Sum of Squares Error
linreg_model_1_RSS<- c(crossprod(linreg_model_1$residuals))

# Calculate Mean Squared error by averaging RSS
linreg_model_1_MSE <- linreg_model_1_RSS / length(linreg_model_1$residuals)

# Calculate Root Mean Squared Error
linreg_model_1_RMSE <- sqrt(linreg_model_1_MSE)
```

Residual Sum of Squares Error: 15939992746048248

Mean Squared Error: 77781862988.9341

Root Mean Squared Error: 278893.999557061

Now, we need to see how the model performs on the testing data. We calculate the predictions and then the R^2 score of these predictions.

```
# predict the results of the model on testing data
linreg_model_1_predictions <- linreg_model_1 %>% predict(test_data)

# calculate the R2 score of the predictions
linreg_model_1_R2 <- R2(linreg_model_1_predictions, test_data$price)
```

R^2 score of the predictions: 0.00685409294336571

Predict Price using multiple columns

Here, we develop a model to predict "Price" using the fields: "year", "manufacturer", "odometer", "condition", "fuel", "transmission", "drive", "size", "type" and "paint_color" and train it.

```
linreg_model_2 <- lm(formula = price ~
                       year+manufacturer+odometer+condition+
                       fuel+transmission+drive+size+type+paint_color,
            data = train_data)
```

The summary of this model can be found by uncommenting the code below. It is commented because it takes a few pages of space.

```
#summary(linreg_model_2)
```

Let's have a look at the performance of this model:

Similar to previous model, we calculate Residual Sum of Squares, Mean Squared Error and Root Mean Squared Error.

```
# calculate Residual Sum of Squares Error
linreg_model_2_RSS<- c(crossprod(linreg_model_2$residuals))

# Calculate Mean Squared Error by averaging RSS
linreg_model_2_MSE <- linreg_model_2_RSS / length(linreg_model_2$residuals)

# Calculate Root Mean Squared Error
linreg_model_2_RMSE <- sqrt(linreg_model_2_MSE)
```

Residual Sum of Squares Error: 15921397485750338

Mean Squared Error: 77691124303.4291

Root Mean Squared Error: 278731.276148604

Now, we need to see how the model performs on the testing data. We calculate the predictions and then the R^2 score of these predictions.

```
# predict the results of the model on testing data
linreg_model_2_predictions <- linreg_model_2 %>% predict(test_data)

# calculate the R2 score of the predictions
linreg_model_2_R2 <- R2(linreg_model_2_predictions, test_data$price)
```

R^2 score of the predictions: 0.0128483216077288

Tuning

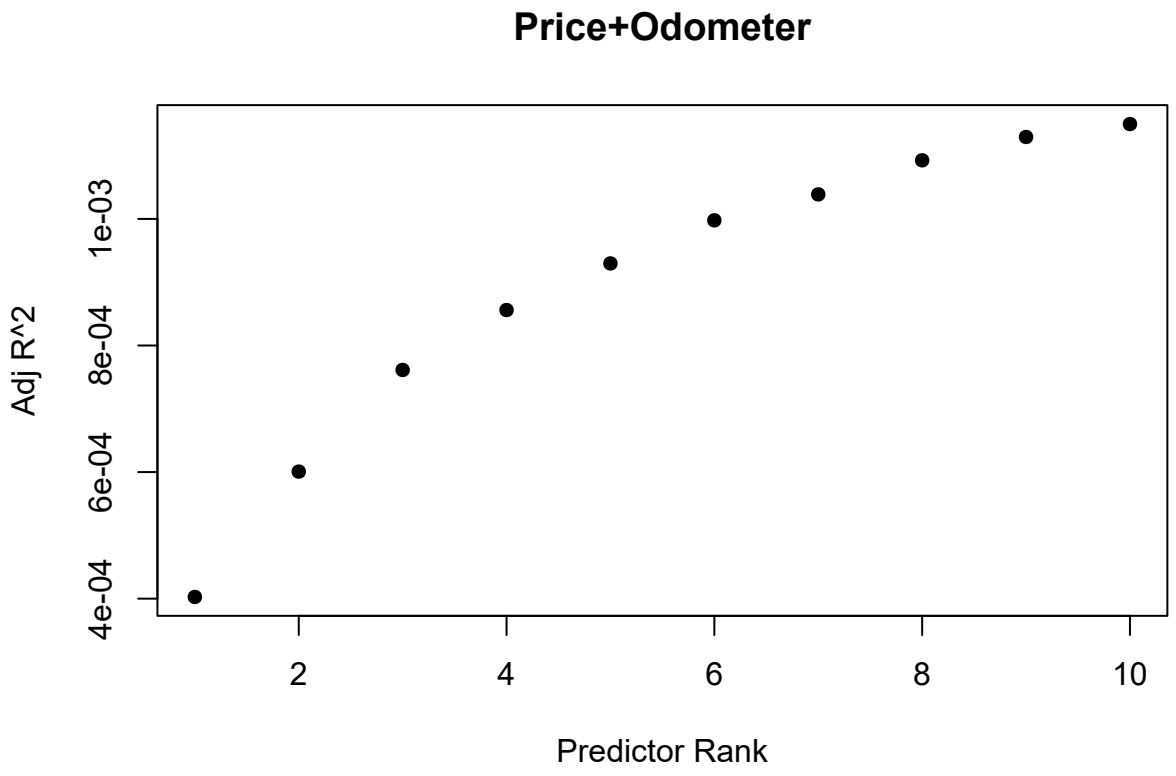

Figure 1: Adjusted R^2 Contribution for Price+Odometer Speeds

Conclusion

Since "Price" is a quantitative value, I decided to predict it using the most commonly looked at factors which are odometer reading and the year of the model since these 2 factors talk about the age of the car and it's usage and age. It can be observed that the errors were very high for this model.

Map Visualization