Exploratory Data Analysis (EDA) for Used Cars data.

Vishal Katti

2024-09-13

Table of contents

List of Figures

List of Tables

Table of Figures

Introduction

After the web-scraping activity is completed, we will now perform exploratory data analysis on the used cars details.

Data Sources

We have 2 CSV files in the **04_CarDetailsConsolidated** which we will ingest.

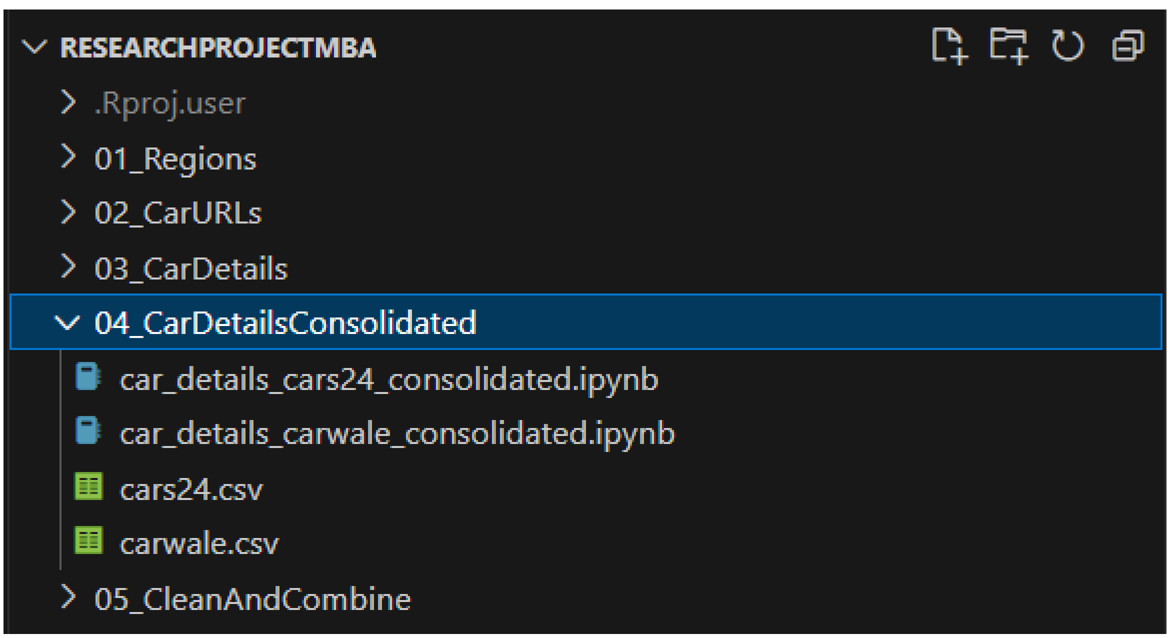

Figure 1: CSV files to be ingested

Ingestion

First we load the required R libraries and create a utility function `f`

```
library(tidyverse)
library(lubridate)  # To manage dates and times
library(janitor)    # To ensure consistent naming and other utilities
library(glue)       # To enable fancy printing

f <- function(x) { print(glue(x))}
```

Since will be cleaning and combining 2 datasets, one from *cars24* and other from *carwale*, we will perform same action twice and handle some special cases in either datasets.

```
cars24 <- read_csv("../04_CarDetailsConsolidated/cars24.csv",
                   na = c("","NA","MISSING","Not Available", "N/A"), show_col_types = FALSE)
carwale <- read_csv("../04_CarDetailsConsolidated/carwale.csv",
                    na = c("","NA","MISSING","Not Available","N/A"), show_col_types = FALSE)
```

> **i** "MISSING" values
>
> In our data extraction scripts from the **03_CarDetails** folder, you would've noticed that while extracting the car attributes, we set the default value to **MISSING** if the value is not available. Thus we know for a fact that **MISSING** indicates a missing value and therefore can be explicitly set to NA while reading the CSV files.
> After a quick manual verification of CSV files, we also understood that some attributes were *Not Available* can can be considered as `NA`.
> The `read_csv` function from **readr** package allows us to explicitly mention which values to be considered as `NA` while reading the data.

Duplicates

First thing we will do is eliminate duplicates. We know there are duplicates since we had to restart our web-scraping scripts couple of times due to various issues.
We will use the `unique` function to remove duplicates.

```
preDup <- nrow(cars24)
cars24 <- unique(cars24)
f("Cars24
-------------------------------------
```

```
Before removing duplicates: {preDup}
After removing duplicates: {nrow(cars24)}
Total duplicates removed: {preDup - nrow(cars24)}")
```

```
Cars24
-------------------------------------
Before removing duplicates: 6392
After removing duplicates: 5889
Total duplicates removed: 503
```

```
preDup <- nrow(carwale)
carwale <- unique(carwale)
f("Carwale
-------------------------------------
Before removing duplicates: {preDup}
After removing duplicates: {nrow(carwale)}
Total duplicates removed: {preDup - nrow(carwale)}")
```

```
Carwale
-------------------------------------
Before removing duplicates: 4764
After removing duplicates: 4355
Total duplicates removed: 409
```

Columns and Data Types

We will now look at the columns and ensure that data types of the columns are correct.

```
str(cars24)
```

```
tibble [5,889 x 18] (S3: tbl_df/tbl/data.frame)
 $ make              : chr [1:5889] "Hyundai" "Renault" "Nissan" "Hyundai" ...
 $ model             : chr [1:5889] "Creta" "Kwid" "Micra" "Eon" ...
 $ variant           : chr [1:5889] "SX PLUS AT 1.6 PETROL" "1.0 MARVEL IRON MAN EDITION AMT"
 $ year              : num [1:5889] 2017 2018 2017 2017 2012 ...
 $ transmission      : chr [1:5889] "Automatic" "Automatic" "Automatic" "Manual" ...
 $ bodyType          : chr [1:5889] "SUV" "Hatchback" "Hatchback" "Hatchback" ...
 $ fuelType          : chr [1:5889] "Petrol" "Petrol" "Petrol" "Petrol" ...
 $ ownerNumber       : num [1:5889] 1 1 2 2 1 1 1 1 2 2 ...
```

```
 $ odometerReading  : num [1:5889] 98493 19178 35474 33963 64557 ...
 $ cityRto          : chr [1:5889] "KA03" "KA03" "KA03" "KA05" ...
 $ listingPrice     : num [1:5889] 973000 407000 528000 381000 463000 ...
 $ fitnessUpto      : chr [1:5889] "29-Feb-2032" "01-Jun-2033" "06-Nov-2032" "15-Jun-2032" .
 $ insuranceType    : chr [1:5889] "Comprehensive" "Insurance Expired" "Comprehensive" "3rd P
 $ duplicateKey     : logi [1:5889] FALSE FALSE NA NA TRUE FALSE ...
 $ city             : chr [1:5889] "Bangalore" "Bangalore" "Bangalore" "Bangalore" ...
 $ registrationYear : num [1:5889] 2017 2018 2017 2017 2012 ...
 $ registrationMonth: num [1:5889] 3 6 11 6 4 7 11 11 6 10 ...
 $ websiteUrl       : chr [1:5889] "https://cars24.com/buy-used-Hyundai-Creta-2017-cars-Banga
```

In above, we can see that `fitnessUpto` column has character datatype instead of date. We also note that the utility of the `websiteUrl` column is complete and we can remove this column from the dataset.

```r
cars24 <- cars24 |>
  select(-websiteUrl) |>
  mutate(fitnessUpto  = dmy(fitnessUpto))

str(cars24)
```

```
tibble [5,889 x 17] (S3: tbl_df/tbl/data.frame)
 $ make             : chr [1:5889] "Hyundai" "Renault" "Nissan" "Hyundai" ...
 $ model            : chr [1:5889] "Creta" "Kwid" "Micra" "Eon" ...
 $ variant          : chr [1:5889] "SX PLUS AT 1.6 PETROL" "1.0 MARVEL IRON MAN EDITION AMT"
 $ year             : num [1:5889] 2017 2018 2017 2017 2012 ...
 $ transmission     : chr [1:5889] "Automatic" "Automatic" "Automatic" "Manual" ...
 $ bodyType         : chr [1:5889] "SUV" "Hatchback" "Hatchback" "Hatchback" ...
 $ fuelType         : chr [1:5889] "Petrol" "Petrol" "Petrol" "Petrol" ...
 $ ownerNumber      : num [1:5889] 1 1 2 2 1 1 1 1 2 2 ...
 $ odometerReading  : num [1:5889] 98493 19178 35474 33963 64557 ...
 $ cityRto          : chr [1:5889] "KA03" "KA03" "KA03" "KA05" ...
 $ listingPrice     : num [1:5889] 973000 407000 528000 381000 463000 ...
 $ fitnessUpto      : Date[1:5889], format: "2032-02-29" "2033-06-01" ...
 $ insuranceType    : chr [1:5889] "Comprehensive" "Insurance Expired" "Comprehensive" "3rd P
 $ duplicateKey     : logi [1:5889] FALSE FALSE NA NA TRUE FALSE ...
 $ city             : chr [1:5889] "Bangalore" "Bangalore" "Bangalore" "Bangalore" ...
 $ registrationYear : num [1:5889] 2017 2018 2017 2017 2012 ...
 $ registrationMonth: num [1:5889] 3 6 11 6 4 7 11 11 6 10 ...
```

```
str(carwale)
```

```
tibble [4,355 x 18] (S3: tbl_df/tbl/data.frame)
 $ make             : chr [1:4355] "Mercedes-Benz" "BMW" "Mercedes-Benz" "BMW" ...
 $ model            : chr [1:4355] "E-Class [2017-2021]" "6 Series GT" "GLA [2017-2020]" "X7
 $ variant          : chr [1:4355] "E 200 Exclusive [2019-2019]" "630i M Sport [2021-2023]" "
 $ year             : num [1:4355] 2023 2021 2019 2021 2013 ...
 $ transmission     : chr [1:4355] "Automatic - 9 Gears, Paddle Shift, Sport Mode" "Automatic
 $ bodyType         : chr [1:4355] "Sedan" "Sedan" "SUV" "SUV" ...
 $ fuelType         : chr [1:4355] "Petrol" "Petrol" "Petrol" "Diesel" ...
 $ ownerNumber      : chr [1:4355] "First" "First" "First" "First" ...
 $ odometerReading  : num [1:4355] 1630 29000 25000 37500 82000 ...
 $ cityRto          : chr [1:4355] "DL14C0018" "GJ1233444" "MH0000000" "HR269200" ...
 $ listingPrice     : num [1:4355] 7350000 6500000 2750000 9800000 555000 ...
 $ fitnessUpto      : logi [1:4355] NA NA NA NA NA NA ...
 $ insuranceType    : chr [1:4355] NA "Comprehensive" "Expired" "Expired" ...
 $ duplicateKey     : logi [1:4355] NA NA NA NA NA NA ...
 $ city             : chr [1:4355] "Delhi" "Delhi" "Delhi" "Delhi" ...
 $ registrationYear : chr [1:4355] "01/01/0001 00:00:00" "01/01/0001 00:00:00" "01/01/0001 00
 $ registrationMonth: logi [1:4355] NA NA NA NA NA NA ...
 $ websiteUrl       : chr [1:4355] "https://www.carwale.com/used-cars/delhi/mercedes-benz-e-c
```

In above, we can see that `fitnessUpto` column has character datatype instead of date. We also note that the utility of the `websiteUrl` column is complete and we can remove this column from the dataset. Additionally, the `ownerNumber`, `registrationYear` and `registrationMonth` columns do not match the data type in the `cars24` dataset. Let's fix this.

```
carwale <- carwale |>
  select(-websiteUrl) |>
  mutate(fitnessUpto  = dmy(fitnessUpto),
         registrationYear  = as_date(registrationYear, format = "%d/%m/%Y %H:%M:%S"),
         registrationMonth = month(registrationYear),
         registrationYear = year(registrationYear),
         registrationYear = if_else(registrationYear == 1, NA_integer_, registrationYear),
         registrationMonth = if_else(is.na(registrationYear), NA_integer_, registrationMonth)
         ownerNumber = case_match(ownerNumber,
                                  "First" ~ 1,
                                  "Second" ~ 2,
                                  "Third" ~ 3,
                                  c("Fourth", "4 or More") ~ 4,
                                  .default = NA_integer_)
```

```r
        )

str(carwale)
```

```
tibble [4,355 x 17] (S3: tbl_df/tbl/data.frame)
 $ make             : chr [1:4355] "Mercedes-Benz" "BMW" "Mercedes-Benz" "BMW" ...
 $ model            : chr [1:4355] "E-Class [2017-2021]" "6 Series GT" "GLA [2017-2020]" "X7
 $ variant          : chr [1:4355] "E 200 Exclusive [2019-2019]" "630i M Sport [2021-2023]" "
 $ year             : num [1:4355] 2023 2021 2019 2021 2013 ...
 $ transmission     : chr [1:4355] "Automatic - 9 Gears, Paddle Shift, Sport Mode" "Automati
 $ bodyType         : chr [1:4355] "Sedan" "Sedan" "SUV" "SUV" ...
 $ fuelType         : chr [1:4355] "Petrol" "Petrol" "Petrol" "Diesel" ...
 $ ownerNumber      : num [1:4355] 1 1 1 1 1 2 1 1 2 1 ...
 $ odometerReading  : num [1:4355] 1630 29000 25000 37500 82000 ...
 $ cityRto          : chr [1:4355] "DL14C0018" "GJ1233444" "MH0000000" "HR269200" ...
 $ listingPrice     : num [1:4355] 7350000 6500000 2750000 9800000 555000 ...
 $ fitnessUpto      : Date[1:4355], format: NA NA ...
 $ insuranceType    : chr [1:4355] NA "Comprehensive" "Expired" "Expired" ...
 $ duplicateKey     : logi [1:4355] NA NA NA NA NA NA ...
 $ city             : chr [1:4355] "Delhi" "Delhi" "Delhi" "Delhi" ...
 $ registrationYear : num [1:4355] NA NA NA NA NA ...
 $ registrationMonth: num [1:4355] NA NA NA NA NA NA 12 12 6 NA ...
```

Comparing the datatypes of the two datasets using waldo::compare function confirms that both datasets have same columns and datatypes and thus, can be combined together for futher clean-up.

```r
waldo::compare(sapply(cars24, typeof), sapply(carwale,typeof))
```

```
v No differences
```

```r
waldo::compare(sapply(cars24, class), sapply(carwale,class))
```

```
v No differences
```

```r
waldo::compare(colnames(cars24), colnames(carwale))
```

```
v No differences
```

Combined Data

Let us combine the 2 datasets.

```r
usedcars <- bind_rows(cars24, carwale)

f("Number of Rows: {nrow(usedcars)}")
```

```
Number of Rows: 10244
```

Column-wise Analysis and clean-up

```r
cat(names(usedcars), sep = "\n")
```

```
make
model
variant
year
transmission
bodyType
fuelType
ownerNumber
odometerReading
cityRto
listingPrice
fitnessUpto
insuranceType
duplicateKey
city
registrationYear
registrationMonth
```

make

Car24 has 15 values for `make` column whereas Carwale has 35 values. They are as follows:

```r
unique(cars24$make)
```

```
 [1] "Hyundai"    "Renault"    "Nissan"     "Maruti"     "Honda"     
 [6] "Tata"       "Toyota"     "KIA"        "Mahindra"   "Datsun"    
[11] "Skoda"      "Ford"       "Volkswagen" "MG"         "Jeep"      
```

```
unique(carwale$make)
```

```
 [1] "Mercedes-Benz" "BMW"           "Skoda"         "Renault"      
 [5] "Toyota"        "Kia"           "Mahindra"      "Maruti Suzuki"
 [9] "Volvo"         "Honda"         "Ford"          "Volkswagen"   
[13] "Land Rover"    "Hyundai"       "MG"            "Jeep"         
[17] "Tata"          "Audi"          "Nissan"        "MINI"         
[21] "Datsun"        "Jaguar"        "BYD"           "Porsche"      
[25] "Lexus"         "Citroen"       "Isuzu"         "Mitsubishi"   
[29] "Force Motors"  "Lamborghini"   "Chevrolet"     "Aston Martin" 
[33] "Fiat"          "Ssangyong"     "Bentley"      
```

Observations

- Even though Carwale has fewer records, it has more variation in terms of `make` of the used cars.
- A few `make` values like KIA, Maruti Suzuki need to be made similar to the corresponding values in Cars24 dataset.

Data Cleaning

```
usedcars <- usedcars |>
  mutate(make = case_match(make,
                           "Kia" ~ "KIA",
                           "Maruti Suzuki" ~ "Maruti",
                           .default = make))
```

A quick look at the frequency of `make` values

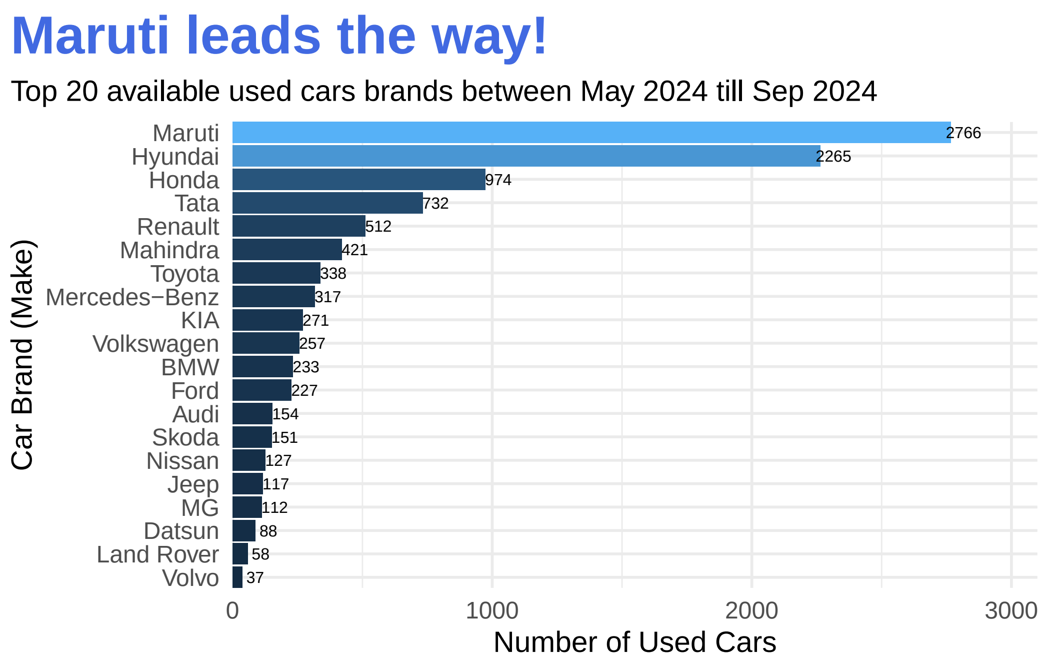

Figure 2: Car Brands (Make)

model

Car24 has 118 values for `model` column whereas Carwale has 552 values.

Observations

- The `model` values in Carwale includes the start and end year of the model. We can remove this to make it more consistent with Cars24 data. There are also a few corrections (Letter case related) which we will perform here.

Data Cleaning

```
usedcars <- usedcars |>
  mutate(model = str_remove(model," ?\\[.*\\]$"),
         lmodel  = str_to_lower(model))

model_bodytype_mapping <- read_csv("multi_bodyTypes.csv", show_col_types = FALSE)

usedcars <- usedcars |>
```

```
left_join(model_bodytype_mapping, by = c("make", "lmodel")) |>
mutate(model = if_else(!is.na(correctmodel), correctmodel, model),
       bodyType = if_else(!is.na(correctbodyType), correctbodyType, bodyType),
       bodyType = if_else(is.na(bodyType), correctbodyType, bodyType)) |>
select(-correctmodel, -correctbodyType, -lmodel)
```

A quick look at the top 20 most famous car `models`

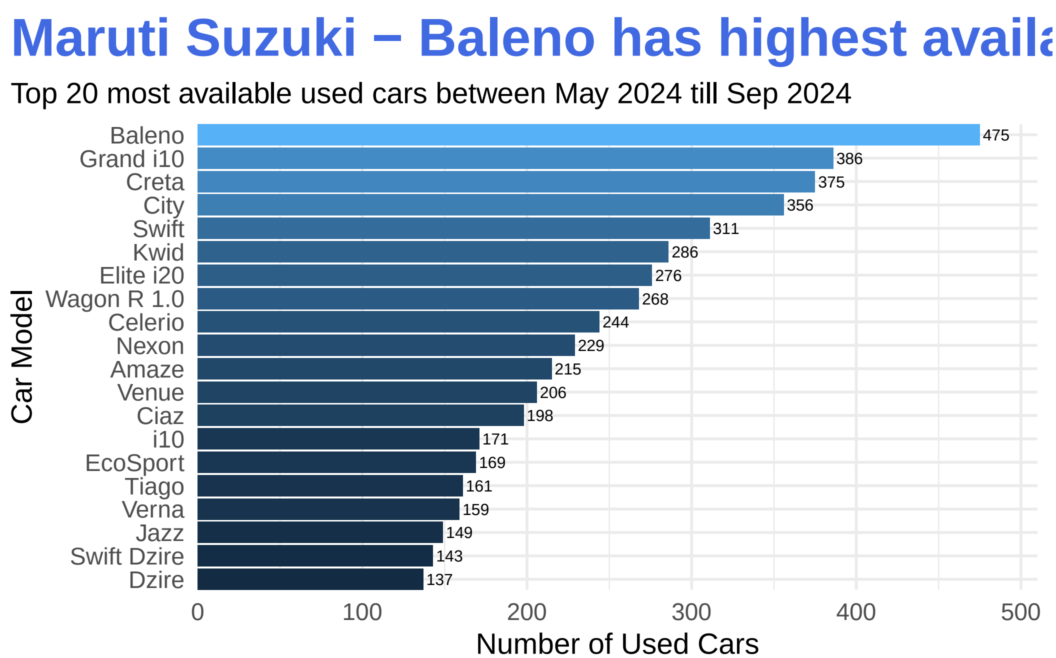

Figure 3: Car Models

variant

Car24 has 908 values for `variant` column whereas Carwale has 1523 values.

Observations

- The `variant` values in Carwale includes the start and end year of the model. We can remove this to make it more consistent with Cars24 data
- The `variant` value sometimes indicate the Engine Capacity, Transmission Type and Fuel Type.

Data Cleaning

No need for any cleaning in `variant` column.

A quick look at the top 20 most famous car `variants`

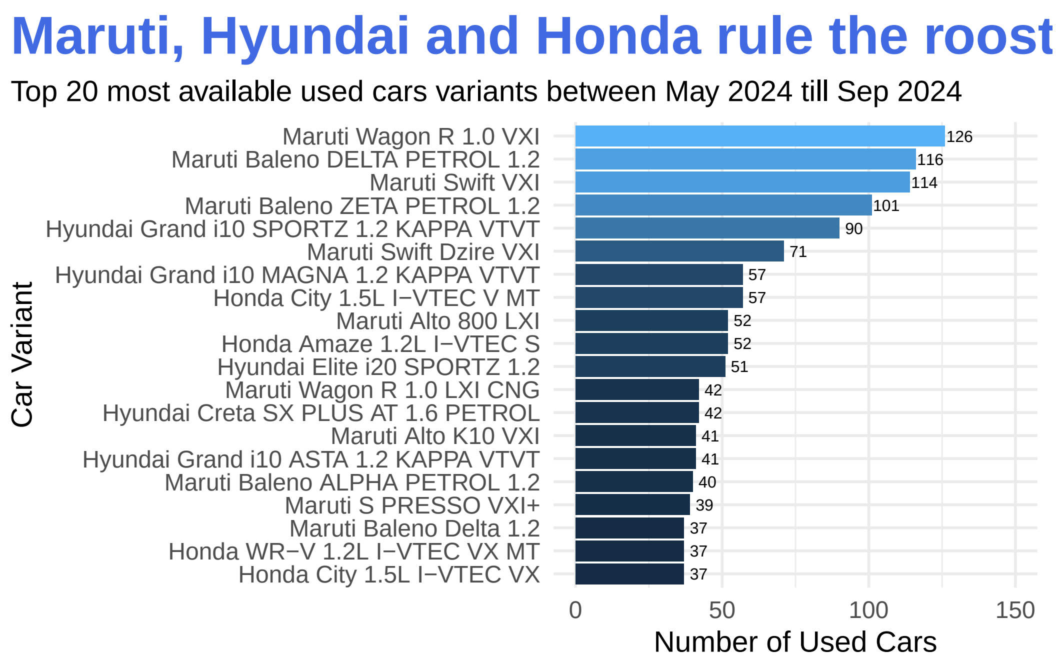

Figure 4: Car Variants

year

Car24 has 15 values for `year` column whereas Carwale has 23 values. They are as follows:

```r
sort(unique(cars24$year))
```

```
 [1] 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024
```

```r
sort(unique(carwale$year))
```

```
 [1] 2000 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016
[16] 2017 2018 2019 2020 2021 2022 2023 2024
```

Observations

- Even though Carwale has fewer records, it has more variation in terms of `year` of the used cars.

Data Cleaning

No data cleaning required for `year` column.

A quick look at the frequency of `year` values

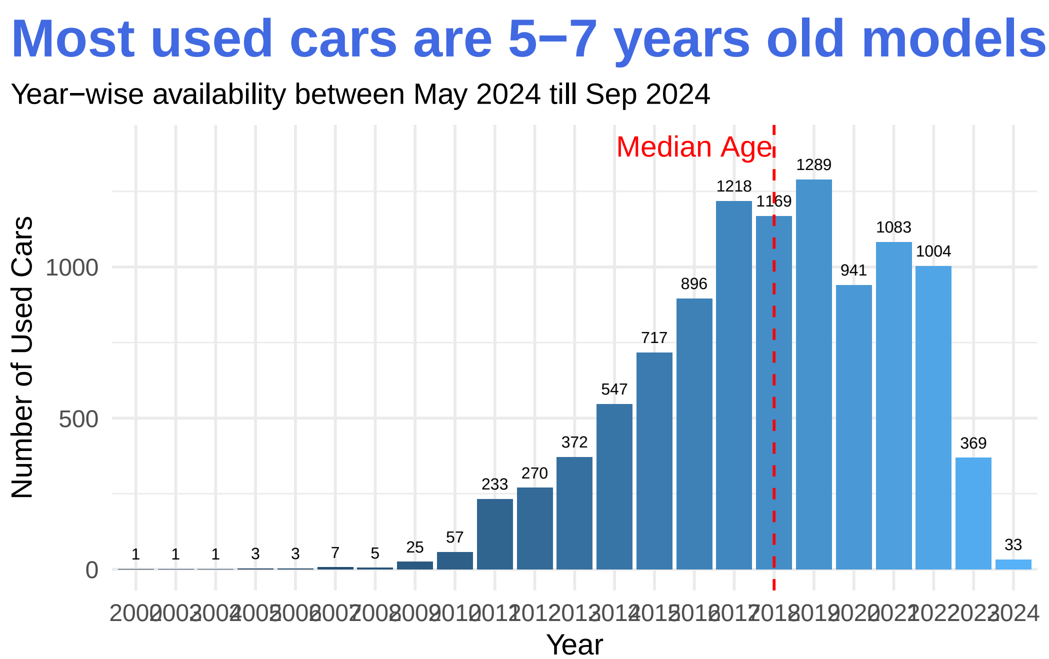

Figure 5: Year of the Car Model/Variant

transmission

Car24 has 2 values for `transmission` column whereas Carwale has 119 values.

Observations

- Even though Carwale has fewer records, it has more variation in terms of `trasmission` of the used cars.
- Additional information about the type of transmission in Automatic is available in Carwale dataset but not available in Cars24.

Data Cleaning

```r
usedcars <- usedcars |>
  mutate(transmission = case_when(str_detect(transmission, "Automatic|AMT") ~ "Automatic",
                                  .default = "Manual"))
```

A quick look at the frequency of `transmission` values

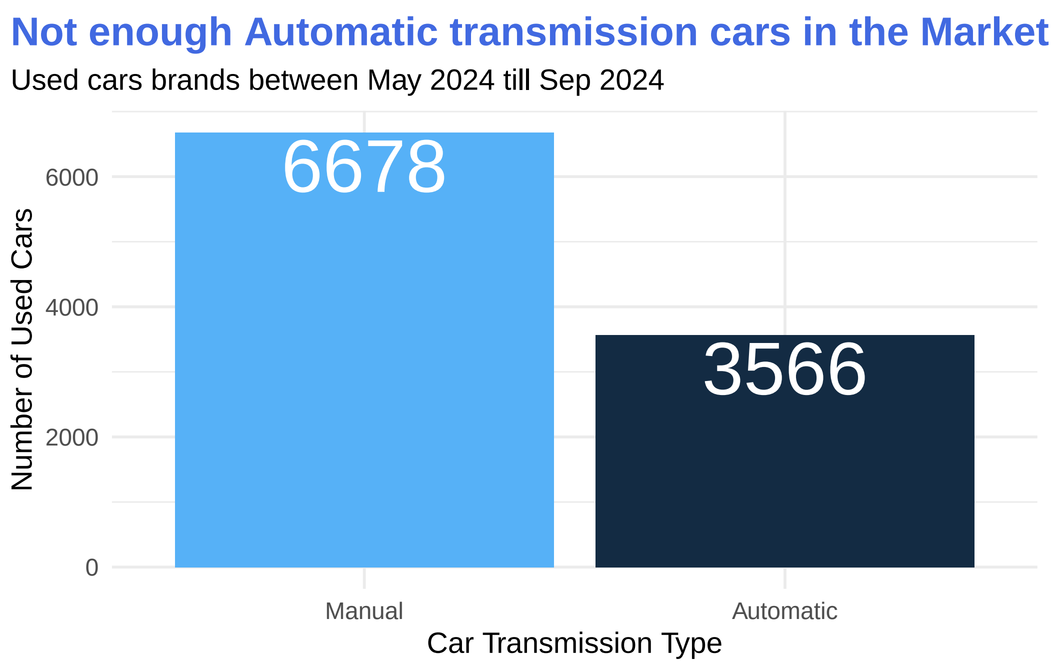

Figure 6: Car Transmission

bodyType

Car24 has 3 values for `bodyType` column whereas Carwale has 12 values. They are as follows:

```r
unique(cars24$bodyType)
```

```
[1] "SUV"       "Hatchback" "Sedan"
```

```r
unique(carwale$bodyType)
```

```
 [1] "Sedan"         "SUV"           "MUV"           "Hatchback"
 [5] "Compact SUV"   "Compact Sedan" NA              "Coupe"
 [9] "Truck"         "Convertible"   "Minivan/Van"   "Minivan"
```

Observations

- There are very few `NA` values which we will impute manually.

Data Cleaning

```
usedcars <- usedcars |>
  mutate(bodyType = case_match(model,
                               "Verito" ~ "Compact Sedan",
                               c("Accord","City Hybrid eHEV") ~ "Sedan",
                               "Fiesta" ~ "Sedan",
                               "Countryman" ~ "Compact SUV",
                               .default = bodyType))
```

A quick look at the frequency of `bodyType` values

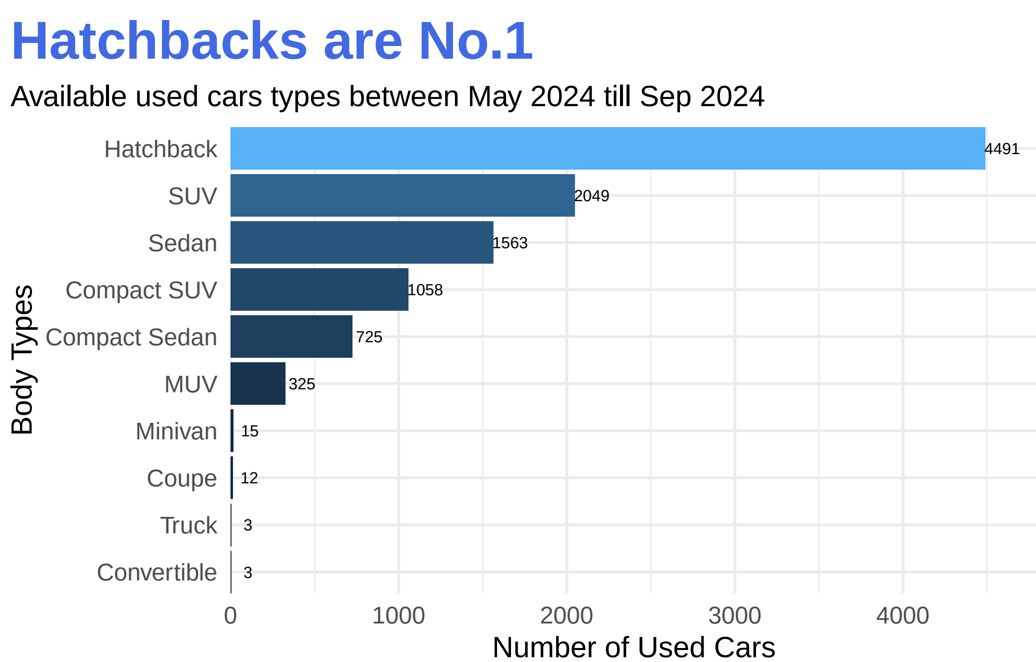

Figure 7: Car Body Type

fuelType

Car24 has 4 values for `fuelType` column whereas Carwale has 14 values. They are as follows:

```r
unique(cars24$fuelType)
```

```
[1] "Petrol"   "Diesel"   "CNG"      "Electric"
```

```r
unique(carwale$fuelType)
```

```
 [1] "Petrol"                                "Diesel"
 [3] "Mild Hybrid(Electric + Petrol)"        "Electric"
 [5] "Hybrid (Electric + Petrol)"            NA
 [7] "LPG"                                   "CNG"
 [9] "CNG + CNG"                             "Petrol + CNG"
[11] "Mild Hybrid (Electric + Diesel)"       "Plug-in Hybrid (Electric + Petrol)"
[13] "Diesel + LPG"                          "Diesel + CNG"
```

Observations

- Even though Carwale has fewer records, it has more variation in terms of `fuelType` of the used cars.

Data Cleaning

A few cars have `NA` values for fuelType column. We will use an external file to map the correct fuelType values. For the sake of simplicity, we will combine a few of these fuel types into groups.

```r
fueltype_mapping <- read_csv("fuelType.csv", show_col_types = FALSE) |> rename("correctfuelTy
usedcars <- usedcars |>
  left_join(fueltype_mapping, by = c("make", "model", "variant")) |>
  mutate(fuelType = coalesce(fuelType, correctfuelType),
         fuelType = case_when(
           str_detect(fuelType, "Hybrid ?\\(Electric") ~ "Electric Hybrid",
           str_detect(fuelType, " \\+ ") ~ "Flex Fuel",
           .default = fuelType
         )) |>
  select(-correctfuelType)
```

A quick look at the frequency of `fuelType` values

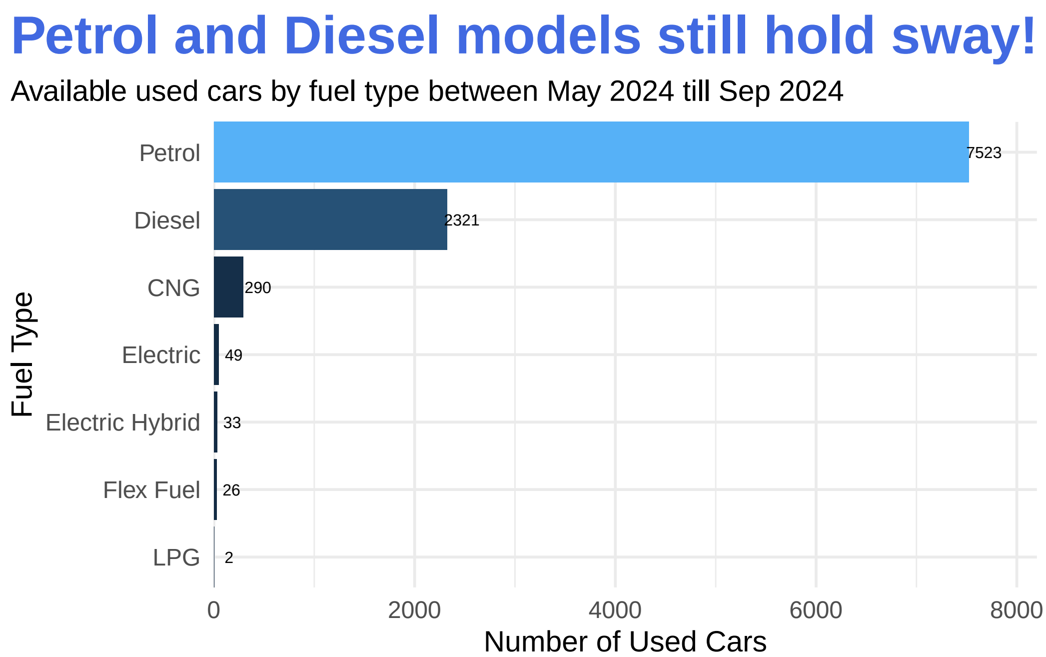

Figure 8: Car Fuel Type

ownerNumber

Car24 has 3 values for `ownerNumber` column whereas Carwale has 5 values. They are as follows:

```
unique(cars24$ownerNumber)
```

```
[1] 1 2 3
```

```
unique(carwale$ownerNumber)
```

```
[1]  1  2  3 NA  4
```

Observations

- Even though Carwale has fewer records, it has more variation in terms of `ownerNumber` of the used cars.
- There are 60 missing values for `ownerNumber` column.

Data Cleaning

There are 60 missing values for `ownerNumber` column. We will impute them in such a way that overall proportion remains the same.

```
non_missing_owners <- usedcars |> filter(!is.na(ownerNumber)) |> nrow()

props <- usedcars |> filter(!is.na(ownerNumber)) |> summarise(p = n()/non_missing_owners, .by

na_indices <- which(is.na(usedcars$ownerNumber))

new_owner_numbers <- sample(
  props$ownerNumber,
  length(na_indices),
  replace = TRUE,
  prob = props$p / sum(props$p)
)

usedcars$ownerNumber[na_indices] <- new_owner_numbers
```

A quick look at the frequency of `ownerNumber` values

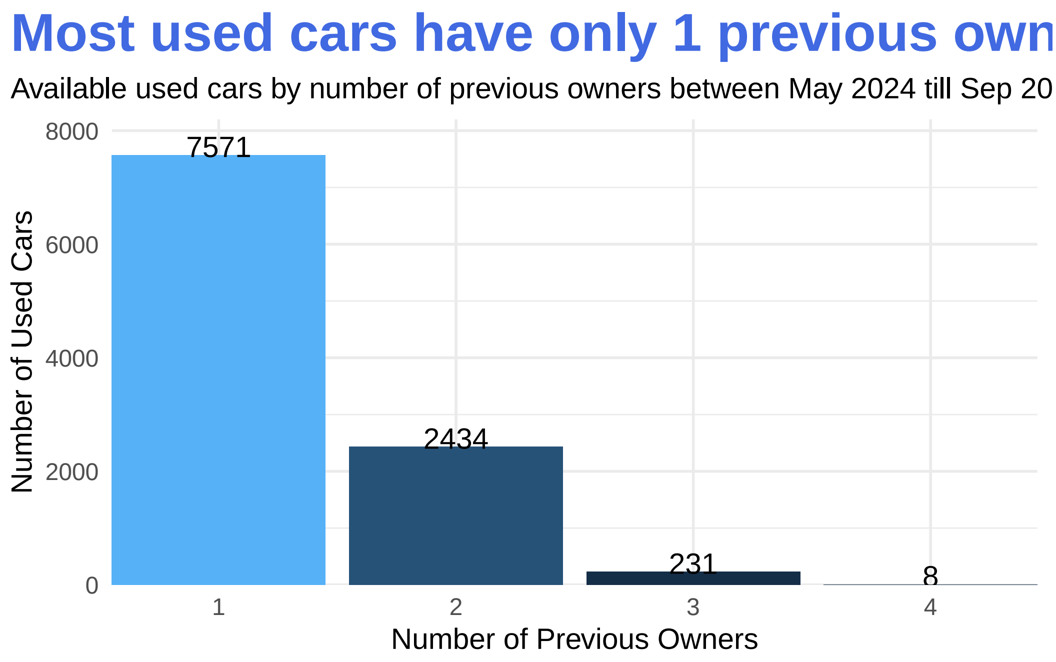

Figure 9: Number of Previous Owners

odometerReading

`odometerReading` is the distance the used car has travelled in kilometers.

Observations

- Even though Carwale has fewer records, it has more variation in terms of `ownerNumber` of the used cars.
- There are 60 missing values for `ownerNumber` column.

Data Cleaning

There are 60 missing values for `ownerNumber` column. We will impute them in such a way that overall proportion remains the same.

```
non_missing_owners <- usedcars |> filter(!is.na(ownerNumber)) |> nrow()

props <- usedcars |> filter(!is.na(ownerNumber)) |> summarise(p = n()/non_missing_owners, .by

na_indices <- which(is.na(usedcars$ownerNumber))

new_owner_numbers <- sample(
  props$ownerNumber,
  length(na_indices),
  replace = TRUE,
  prob = props$p / sum(props$p)
)

usedcars$ownerNumber[na_indices] <- new_owner_numbers
```

A quick look at the frequency of `ownerNumber` values

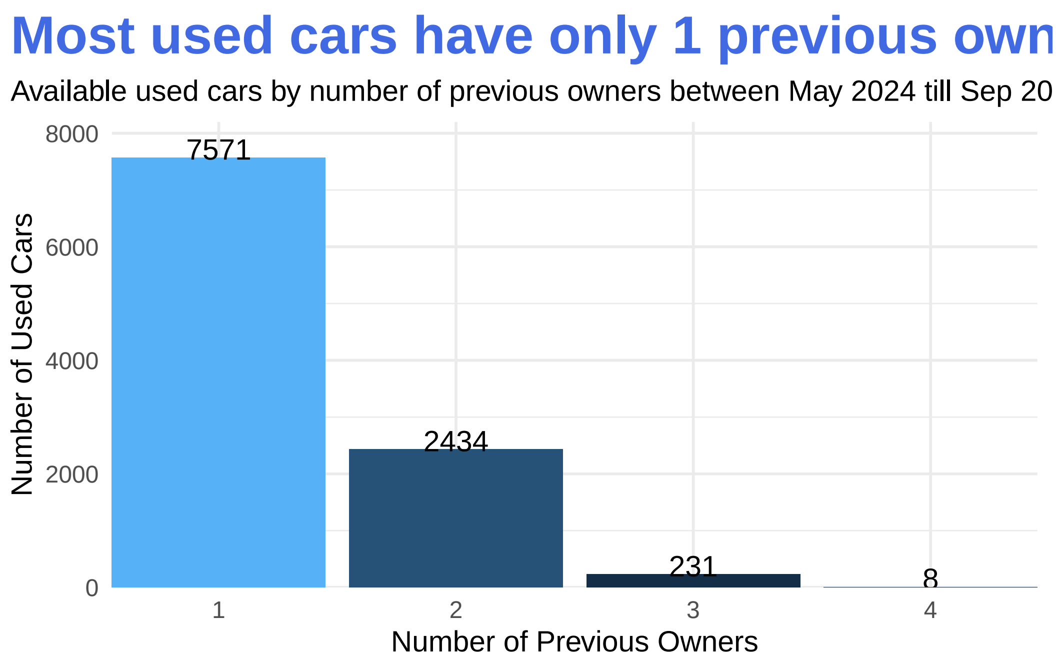

Figure 10: Distance Travelled