Used_Car_Lab_1_DataVisualization

January 24, 2021

1 Lab 1 - Used Cars in the USA

By: David Wei, Sophia Wu, Dhruba Dey, Queena Wang

1.0.1 Business Understanding:

As a team, we decided to work on a dataset which can easily be understood from our common life experience without needing specific domain knowledge such as finance, marketing and so forth. Secondly, we wanted to dabble ourselves with a dataset, that we had not previously exposed to, such as Real Estate, Life expectancy, etc. Off course, we must meet the project guidelines of 30,000 rows and 10 features, which in fact prompted us to look for somewhat large dataset with multiple features so that can effectively cull right set of features having numerical, boolean and categorical values. In addition, working with a large dataset would provide us with the experience that we have not acquired yet. This thought process of data selection has led us to grab "Used Car Dataset", which has 3 million rows and 66 features. We wanted to build 3 different types of models on this dataset. First is Regression, second binary classification using Logistic Regression and third multiple classification using either K-nearest or K-means Random Forest depending on the best fit. We picked "Price" as the response feature for Regression model because it has numerical value and a pivotal point in the used car purchase. For Logistic Regression, we opted for "has_accidents" as the response variable because it has boolean values and an interesting point to observe whether we can effectively classify the used car with accidents. Finally, we used "body_type" as the target variable for multi-value classification. "Body_type" has 9 attributes which we believed a good candidate for multi-class classification. We applied first our domain knowledge, which is our collective experience of used car buying, to mine features that we thought would be relevant for making decision on a used car purchase. That helped us reduce the numbers of features from 66 to 40. Then we took a close at each feature and tried to eliminate as many as possibleto keep the data size manageable so that we could handle data wrangling and model building, to be performed in the next phase, with the computing resources available to us. Also, we provided the justification as to why a feature was removed from the list of 40. However, we were cautious about categorical variable because itwould spawn to multiple variables after encoding. That is the key reason why we restricted the categorical feature to one. In the final dataset, we have 19 variables comprising 15 numerical, 1 boolean and 3 categorical. By the way, removing the missing values after sub-setting the data set to the intended features, we reduced the numbers of rows close to 700K from 3M. And we would also conduct PCA and Regularizations for dimensionality reduction. Based on the analysis and understanding of the data, we believed we selected the right set of data for the purpose of this project.

The general approach for the project is to build the model on the training data and cross-validate the model on the test data to find it effectiveness. We would use cross-validation technics such as k-fold etc. as appropriate. As noted earlier, we will use the Regression technic for "Price" prediction. And

the effectiveness measures to be used to validate the Regression model are RMSE and CV Press. The Area Under the ROC curve (AUC) is an aggregated metric that evaluates how well a logistic regression model classifies positive and negative outcomes at all possible cutoffs. Our objective is to find the right cut-off to maximize the area under ROC curve (AUC). Off course, we will present the confusion matrix to know the misclassification rate, recall and precision. Similarly, for multi-class classification, we will use accuracy, recall and precision matrix to evaluate the effectiveness of the model. Also, we will keep an eye on sensitivity and specificity so that we do not want to build a model which is very either sensitive or specific.

[1]:
```
#!pip install missingno
#!pip install plotnine
#pip install ptitprince
```

[2]:
```
#importing libraries and reading in file
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore') #ignoring warnings

import missingno as msno
import matplotlib.pyplot as plt
import seaborn as sns
from plotnine.data import economics
from plotnine import ggplot, aes, geom_line

from scipy.stats import trim_mean, kurtosis
from scipy.stats.mstats import mode, gmean, hmean
import ptitprince as pt

import sklearn.preprocessing as preprocessing
import sklearn.model_selection as cross_validation
import sklearn.linear_model as linear_model
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
```

NOTE: need to change file per user

[3]:
```
df_raw = pd.read_csv('data/rawdata.csv')
#df_raw = pd.read_csv('https://raw.githubusercontent.com/chee154/
 ↪ml-Py-used_cars/main/data/kaggle_used_cars_data.csv')
#df_raw = pd.read_csv('C:/Users/Queen/Documents/GitHub/ml-Py-used_cars/data/
 ↪kaggle_used_cars_data.csv')
```

[4]:
```
df_raw.head(5)
```

[4]:
```
                 vin back_legroom  bed bed_height bed_length        body_type  \
0  ZACNJABB5KPJ92081      35.1 in  NaN        NaN        NaN  SUV / Crossover
1  SALCJ2FX1LH858117      38.1 in  NaN        NaN        NaN  SUV / Crossover
2  JF1VA2M67G9829723      35.4 in  NaN        NaN        NaN            Sedan
3  SALRR2RV0L2433391      37.6 in  NaN        NaN        NaN  SUV / Crossover
4  SALCJ2FXXLH862327      38.1 in  NaN        NaN        NaN  SUV / Crossover

  cabin      city  city_fuel_economy  combine_fuel_economy  …  transmission  \
0   NaN   Bayamon                NaN                   NaN  …             A
1   NaN  San Juan                NaN                   NaN  …             A
2   NaN  Guaynabo               17.0                   NaN  …             M
3   NaN  San Juan                NaN                   NaN  …             A
4   NaN  San Juan                NaN                   NaN  …             A

          transmission_display  trimId     trim_name  vehicle_damage_category  \
0  9-Speed Automatic Overdrive  t83804  Latitude FWD                      NaN
1  9-Speed Automatic Overdrive  t86759         S AWD                      NaN
2               6-Speed Manual  t58994          Base                      NaN
3  8-Speed Automatic Overdrive  t86074    V6 HSE AWD                      NaN
4  9-Speed Automatic Overdrive  t86759         S AWD                      NaN

  wheel_system wheel_system_display wheelbase    width  year
0          FWD    Front-Wheel Drive  101.2 in  79.6 in  2019
1          AWD      All-Wheel Drive  107.9 in  85.6 in  2020
2          AWD      All-Wheel Drive  104.3 in  78.9 in  2016
3          AWD      All-Wheel Drive    115 in  87.4 in  2020
4          AWD      All-Wheel Drive  107.9 in  85.6 in  2020

[5 rows x 66 columns]
```

[5]:
```
print("Total # of Records: " + str(df_raw.shape[0]))
print("Total # of Columns: " + str(df_raw.shape[1]))
```

```
Total # of Records: 3000040
Total # of Columns: 66
```

1.0.2 Data Meaning Type

Domain based Attribute Reduction Before we do a deep dive into the types of our data, we will first look into reducing it from a The total dataset has 66 attributes After a quick observation of the column headers, we can deduce that not all columns will be necessary for our analysis. Reasons for removing them below:

subsetting columns by referencing the column indexes

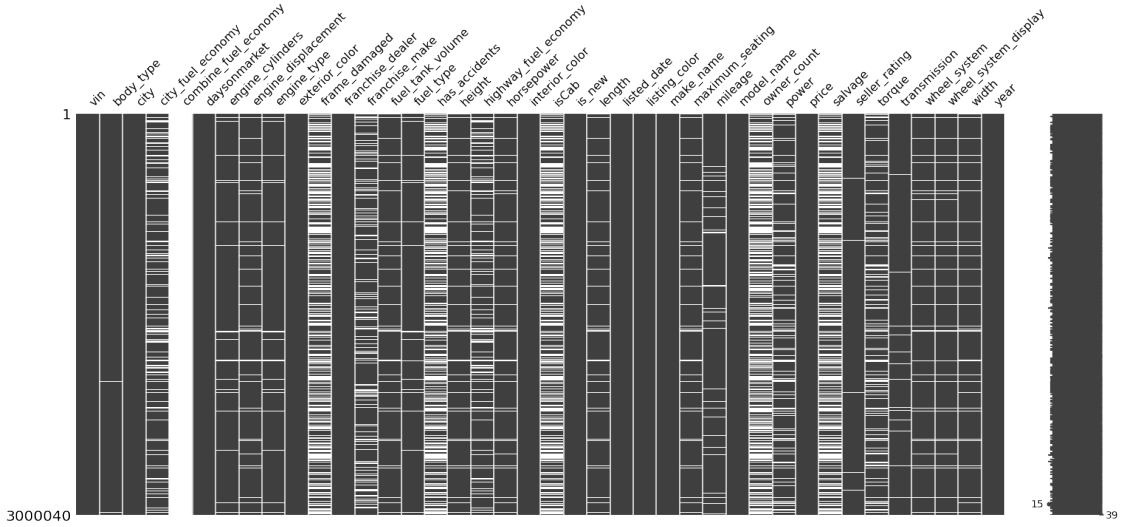
[6]:
```
df_cln_1 = df_raw.iloc[:, np.r_[0,5,7:11,13:17,18:21,22:30,32,35:38,42:50,51,55:
 ↪57,61:63,64:66]]
print(df_cln_1.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000040 entries, 0 to 3000039
Data columns (total 40 columns):
 #   Column                Dtype
---  ------                -----
 0   vin                   object
 1   body_type             object
 2   city                  object
 3   city_fuel_economy     float64
 4   combine_fuel_economy  float64
 5   daysonmarket          int64
 6   engine_cylinders      object
 7   engine_displacement   float64
 8   engine_type           object
 9   exterior_color        object
 10  frame_damaged         object
 11  franchise_dealer      bool
 12  franchise_make        object
 13  fuel_tank_volume      object
 14  fuel_type             object
 15  has_accidents         object
 16  height                object
 17  highway_fuel_economy  float64
 18  horsepower            float64
 19  interior_color        object
 20  isCab                 object
 21  is_new                bool
 22  length                object
 23  listed_date           object
 24  listing_color         object
 25  make_name             object
 26  maximum_seating       object
 27  mileage               float64
 28  model_name            object
 29  owner_count           float64
 30  power                 object
 31  price                 float64
 32  salvage               object
 33  seller_rating         float64
 34  torque                object
 35  transmission          object
 36  wheel_system          object
 37  wheel_system_display  object
 38  width                 object
 39  year                  int64
dtypes: bool(2), float64(9), int64(2), object(27)
memory usage: 875.5+ MB
None
```

1.0.3 Data Quality

[7]:
```
msno.matrix(df_cln_1)
```

[7]: <AxesSubplot:>



After doing a quick profiling on some our identified columsn, we can see that both engine_cylinders and engine_type are the same. Additionally, we also found that the prefixes and suffixes attached to them are descriptive of it and thus not a continuous value.

removing city_fuel_economy since a quick vizualition of our dataset shows that ALL values are empty

[8]:
```
df_cln_1 = df_cln_1.drop(columns='combine_fuel_economy')
print(df_cln_1.shape[1])
```

```
39
```

We also found that 'engine_cylinders','engine_type' contains the same data in two separate columns, so we first test if this condition is true and if it is, we will remove one.

observing if 'engine_cylinders','engine_type' is the same data

[10]:
```
if df_cln_1['engine_cylinders'].equals(df_cln_1['engine_type']) == True:
    df_cln_1 = df_cln_1.drop(columns='engine_cylinders')
    print(df_cln_1.shape[1])
```

```
38
```

We also found additional columns that were simply descriptions of another columns, for example. The 'wheel_system_display' attribute is simply a longer, more descriptive version of the 'wheel_system' attribute ("Front-Wheel Drive" vs "FWD"). For columns that follow this trend, we will remove the descriptive column from our dataset.

[11]:
```
print(df_cln_1[['wheel_system']])
print(df_cln_1[['wheel_system_display']])
df_cln_1 = df_cln_1.drop(columns='wheel_system_display')
print(df_cln_1.shape[1])
```

```
        wheel_system
0                FWD
1                AWD
2                AWD
3                AWD
4                AWD
…                …
3000035          FWD
3000036          FWD
3000037          FWD
3000038          AWD
3000039          FWD

[3000040 rows x 1 columns]
        wheel_system_display
0          Front-Wheel Drive
1            All-Wheel Drive
2            All-Wheel Drive
3            All-Wheel Drive
4            All-Wheel Drive
…                        …
3000035    Front-Wheel Drive
3000036    Front-Wheel Drive
3000037    Front-Wheel Drive
3000038      All-Wheel Drive
3000039    Front-Wheel Drive

[3000040 rows x 1 columns]
37
```

We also discussed that 'interior_color' attribute has 45,726 distinct color values and so due to the sheer volume and complexity (# of levels in this attribute). We also decided to remove it from our dataset since it is unrealistic for the type of modeling we are doing and would effect the overall performance of our model.

#note this plot takes forever to run #(ggplot(df_cln_1)+aes(x="interior_color", y="price")+geom_line())

[12]:
```
print(df_cln_1['interior_color'].nunique())

df_cln_1 = df_cln_1.drop(columns='interior_color')
print(df_cln_1.shape[1])
```

```
45726
```

36

Another attribute we found that could be removed as the "listed_date" attribute. Since there was no other 'datetype' attributes available in our dataset, we didn't really see a point with keeping it as there was no other data reference to use it with. Additionally, the attribute 'daysonmarket' already pre-aggregated the number of days it took to sell a vehicle, which the listed date would've been used for otherwise. Because of this, we will also remove it from our dataframe.

[13]:
```
df_cln_1 = df_cln_1.drop(columns='listed_date')
print(df_cln_1.shape[1])
```

35

At this point, we decided to take a break from subsetting our data and take a look at the amount of missing values in the current dataframe. Our intention in mind was to see what the data looked like after it was cleaned to further proceed reducing the amount of attributes we had.

1.0.4 Data Cleaning - Duplicates, Missing Data, Nulls

Now that our datatypes have been adjusted. We will work on cleaning up any empty data in our dataset. To begin, we will check if any of the VINs have duplicates in the dataset, since this the VIN is unique to a car we are expecting there shouldn't be, but in cases there is, we will remove it.

[14]:
```
df_cln_1[df_cln_1.duplicated(['vin'], keep=False)].sort_values('vin')
```

[14]:

	vin	body_type	city	\
2000032	1C3CCCBG4FN686074	Sedan	Deer Park	
2000052	1C3CCCBG4FN686074	Sedan	Deer Park	
2000043	1C6RR7FT5ES405345	Pickup Truck	Bonners Ferry	
2000023	1C6RR7FT5ES405345	Pickup Truck	Bonners Ferry	
1000038	1FA6P8TH0J5179933	Coupe	Groveport	
…	…	…	…	
1000030	7FART6H88LE013181	SUV / Crossover	Westerville	
2000037	JTEBU5JR5K5687002	SUV / Crossover	Airway Heights	
2000057	JTEBU5JR5K5687002	SUV / Crossover	Airway Heights	
2000044	JTNK4RBE3K3044291	NaN	Colville	
2000024	JTNK4RBE3K3044291	NaN	Colville	

	city_fuel_economy	daysonmarket	engine_displacement	engine_type	\
2000032	19.0	530	3600.0	V6	
2000052	19.0	530	3600.0	V6	
2000043	NaN	49	5700.0	V8	
2000023	NaN	49	5700.0	V8	
1000038	21.0	7	2300.0	I4	
…	…	…	…	…	
1000030	40.0	28	2000.0	I4 Hybrid	
2000037	17.0	20	4000.0	V6	
2000057	17.0	20	4000.0	V6	
2000044	NaN	33	NaN	I4	

```
2000024                NaN            33                   NaN          I4

                exterior_color frame_damaged  franchise_dealer  …  \
2000032                   Blue         False             False  …
2000052                   Blue         False             False  …
2000043                   GRAY         False             False  …
2000023                   GRAY         False             False  …
1000038  Lightning Blue Metallic         False             False  …
…                         …            …                 …  …
1000030     Platinum White Pearl           NaN             False  …
2000037                   WHITE        False             False  …
2000057                   WHITE        False             False  …
2000044                   BLACK        False              True  …
2000024                   BLACK        False              True  …

        owner_count               power    price salvage seller_rating  \
2000032         1.0  295 hp @ 6,350 RPM  13488.0   False      4.453125
2000052         1.0  295 hp @ 6,350 RPM  13488.0   False      4.453125
2000043         1.0  395 hp @ 5,600 RPM  20480.0   False      4.750000
2000023         1.0  395 hp @ 5,600 RPM  20480.0   False      4.750000
1000038         1.0  310 hp @ 5,500 RPM  24000.0   False      4.211765
…             …                   …      …     …           …
1000030         NaN  212 hp @ 6,200 RPM  33970.0     NaN      4.377049
2000037         NaN  270 hp @ 5,600 RPM  44995.0   False      3.964286
2000057         NaN  270 hp @ 5,600 RPM  44995.0   False      3.964286
2000044         1.0                 NaN  21000.0   False      4.583333
2000024         1.0                 NaN  21000.0   False      4.583333

                      torque  transmission wheel_system    width  year
2000032  262 lb-ft @ 4,250 RPM             A          FWD  73.6 in  2015
2000052  262 lb-ft @ 4,250 RPM             A          FWD  73.6 in  2015
2000043  407 lb-ft @ 3,950 RPM           NaN          4WD  79.4 in  2014
2000023  407 lb-ft @ 3,950 RPM           NaN          4WD  79.4 in  2014
1000038  350 lb-ft @ 3,000 RPM             A          RWD  81.9 in  2018
…                        …           …          …       …   …
1000030                  NaN           CVT          AWD    73 in  2020
2000037  278 lb-ft @ 4,400 RPM             A          4WD  75.8 in  2019
2000057  278 lb-ft @ 4,400 RPM             A          4WD  75.8 in  2019
2000044                  NaN             A          NaN      NaN  2019
2000024                  NaN             A          NaN      NaN  2019

[80 rows x 35 columns]
```

We can see that that there are duplicates even though the 'vin' should be specific and distinct to each car. There is a total of 80 records that contain duplicates. We will then remove these duplicates while keeping the 'first' record so that one copy of the duplcites will remain. We can see that after we cleaned these duplicates, our total # of records drop from 3,000,040 to 3,000,000.

[15]:
```
print(df_cln_1.shape[0])
```

3000040

[16]:
```
df_cln_1 = df_cln_1.drop_duplicates(subset=['vin'], keep='first')
print("# of records after removing duplicates: "+str(df_cln_1.shape[0]))
```

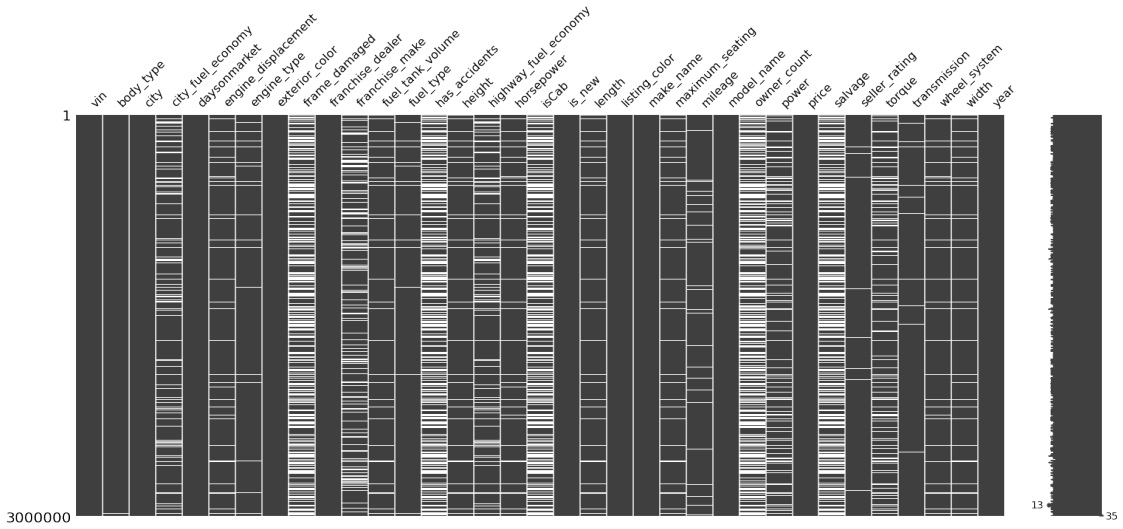
```
# of records after removing duplicates: 3000000
```

After we have cleaned our duplicates. A quick visualization of our data shows that almost every column has empty values. We will focus on analyzing those that have a large amount of empty data (ex. Frame_Damaged, has_accidents, isCab, etc.)

visualizatin of our data BEFORE removing all rows with missing data

[17]:
```
msno.matrix(df_cln_1)
```

[17]: <AxesSubplot:>



remove the missing values

[18]:
```
df_cln_2 = df_cln_1.dropna()
print(len(df_cln_2))
```

697989

checking the row counts of columns to see the missing rows

[19]:
```
df_cln_2.count()
```

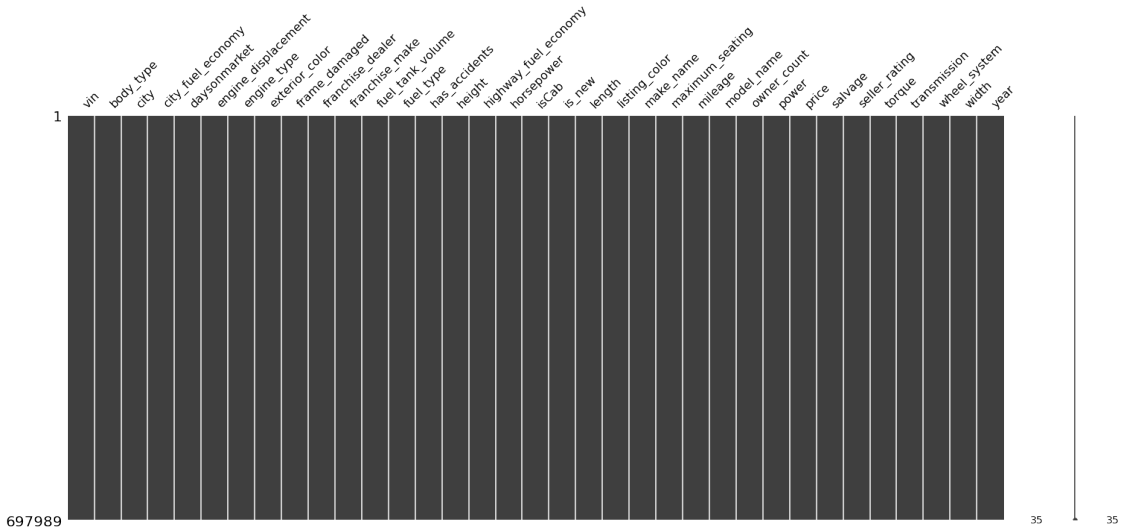
[19]:
```
vin                    697989
body_type              697989
```

```
city                    697989
city_fuel_economy       697989
daysonmarket            697989
engine_displacement     697989
engine_type             697989
exterior_color          697989
frame_damaged           697989
franchise_dealer        697989
franchise_make          697989
fuel_tank_volume        697989
fuel_type               697989
has_accidents           697989
height                  697989
highway_fuel_economy    697989
horsepower              697989
isCab                   697989
is_new                  697989
length                  697989
listing_color           697989
make_name               697989
maximum_seating         697989
mileage                 697989
model_name              697989
owner_count             697989
power                   697989
price                   697989
salvage                 697989
seller_rating           697989
torque                  697989
transmission            697989
wheel_system            697989
width                   697989
year                    697989
dtype: int64
```

visualizatin of our data AFTER removing all rows with missing data

[20]:
```
msno.matrix(df_cln_2)
```

[20]:
```
<AxesSubplot:>
```



We decided that since our original dataset was large (10gb with 3million records), that instead of imputing data based on the mean or other statistical types, that deleting all records with empty values still returned 700,000 total records that not only provides plenty of data leftover, but also raw data that isn't imputed.

1.0.5 Data Cleaning - Datatypes

Obviously at this point we need to convert a few of our data columns to the appropriate data type by removing parts of the value string that we do not need such as "gal" in the ful_tank_volume

Finding all unique values per column to see what values we need to clean

[23]:
```
columns_that_need_cleaning =␣
 ↪['engine_type','exterior_color','frame_damaged','franchise_dealer','franchise_make','fuel_t
for col in columns_that_need_cleaning:
        print(df_cln_2[col].unique())
```

```
['I4' 'V6' 'V8' 'H4' 'I6' 'V8 Flex Fuel Vehicle' 'V6 Flex Fuel Vehicle'
 'I5' 'I4 Diesel' 'I4 Flex Fuel Vehicle' 'I4 Hybrid' 'V6 Diesel'
 'I6 Diesel' 'I3' 'W12 Flex Fuel Vehicle' 'V6 Biodiesel' 'V12' 'V10' 'W12'
 'H6' 'H4 Hybrid' 'V6 Hybrid' 'I4 Compressed Natural Gas' 'W16' 'R2'
 'I6 Hybrid' 'V6 Compressed Natural Gas']
['Silver Ice Metallic' 'Black' 'Diamond Black Crystal Pearlcoat' …
 'VICTORIA BLACK' 'Modern S' 'Cosmic']
[False True]
[ True]
['Chevrolet' 'Jeep' 'Cadillac' 'Chrysler' 'Dodge' 'Kia' 'RAM' 'Mazda'
 'Audi' 'Hyundai' 'Ford' 'Toyota' 'Lincoln' 'Volvo' 'GMC' 'Volkswagen'
 'BMW' 'Lexus' 'Buick' 'Subaru' 'Scion' 'Honda' 'Acura' 'Nissan'
 'INFINITI' 'Porsche' 'Rolls-Royce' 'Lamborghini' 'Bentley'
 'Mercedes-Benz' 'Jaguar' 'Land Rover' 'Maserati' 'Ferrari' 'MINI' 'FIAT'
```

```
 'Alfa Romeo' 'Mitsubishi' 'Aston Martin' 'Lotus' 'McLaren' 'SRT'
 'Genesis' 'smart' 'Shelby' 'Pagani']
['15.8 gal' '17.4 gal' '19.4 gal' '22 gal' '24.6 gal' '13.5 gal'
 '13.2 gal' '15.9 gal' '12.7 gal' '22.5 gal' '21.5 gal' '18.5 gal'
 '26 gal' '12.4 gal' '19.2 gal' '18.6 gal' '23.2 gal' '24 gal' '16.4 gal'
 '13.7 gal' '19 gal' '14.5 gal' '14 gal' '18 gal' '16.9 gal' '21 gal'
 '19.5 gal' '31 gal' '18.8 gal' '16 gal' '16.6 gal' '21.7 gal' '20 gal'
 '15.3 gal' '22.2 gal' '17.2 gal' '21.1 gal' '16.1 gal' '13 gal'
 '23.8 gal' '17 gal' '15.1 gal' '19.8 gal' '23 gal' '20.5 gal' '11.9 gal'
 '14.2 gal' '14.3 gal' '14.8 gal' '17.7 gal' '16.5 gal' '17.8 gal'
 '15.6 gal' '17.1 gal' '15.5 gal' '10.6 gal' '15.7 gal' '18.1 gal'
 '20.1 gal' '12.8 gal' '20.3 gal' '21.6 gal' '21.9 gal' '11.4 gal'
 '36 gal' '16.8 gal' '15.4 gal' '16.3 gal' '28.3 gal' '17.9 gal'
 '27.7 gal' '13.6 gal' '17.5 gal' '28 gal' '27 gal' '22.4 gal' '20.6 gal'
 '26.4 gal' '11.6 gal' '11.8 gal' '12 gal' '19.3 gal' '13.1 gal'
 '10.8 gal' '14.9 gal' '19.1 gal' '12.1 gal' '27.3 gal' '38 gal'
 '11.3 gal' '32 gal' '10 gal' '23.3 gal' '25 gal' '11.1 gal' '16.2 gal'
 '23.5 gal' '15 gal' '33 gal' '23.6 gal' '25.4 gal' '33.5 gal' '17.6 gal'
 '23.7 gal' '25.6 gal' '21.8 gal' '18.4 gal' '10.5 gal' '14.6 gal'
 '20.4 gal' '14.4 gal' '27.6 gal' '12.2 gal' '22.7 gal' '25.1 gal'
 '18.3 gal' '12.6 gal' '18.2 gal' '11 gal' '17.3 gal' '9.2 gal' '30 gal'
 '--' '24.3 gal' '27.8 gal' '28.5 gal' '22.8 gal' '31.5 gal' '34 gal'
 '8.7 gal' '23.9 gal' '29 gal' '18.7 gal' '20.2 gal' '35 gal' '15.2 gal'
 '21.4 gal' '16.7 gal' '39 gal' '40.7 gal' '21.2 gal' '8 gal' '12.3 gal'
 '24.1 gal' '42 gal' '13.3 gal' '7.7 gal' '7.8 gal']
['Gasoline' 'Flex Fuel Vehicle' 'Diesel' 'Hybrid' 'Biodiesel'
 'Compressed Natural Gas']
[False True]
['57.6 in' '55.1 in' '70.7 in' '69.9 in' '69.3 in' '65 in' '64.1 in'
 '58.7 in' '66.5 in' '70.8 in' '70.9 in' '73.6 in' '58.5 in' '73.9 in'
 '65.3 in' '55.7 in' '67.7 in' '64.5 in' '58.3 in' '57.7 in' '71 in'
 '76.3 in' '66.1 in' '54.6 in' '74.4 in' '64.8 in' '55.8 in' '69.4 in'
 '66.3 in' '53.1 in' '66 in' '72.6 in' '56.5 in' '66.2 in' '67.1 in'
 '65.7 in' '70.6 in' '70.2 in' '67 in' '54.4 in' '75.5 in' '58.9 in'
 '74 in' '67.9 in' '57.8 in' '69 in' '52.9 in' '67.3 in' '56.8 in'
 '63.3 in' '58.1 in' '68.9 in' '55.3 in' '66.8 in' '57.4 in' '70.4 in'
 '56.2 in' '54.2 in' '56.3 in' '67.4 in' '55.9 in' '58 in' '57.9 in'
 '57.2 in' '68.3 in' '59.6 in' '54.9 in' '68.5 in' '67.5 in' '68.4 in'
 '57.1 in' '51.8 in' '76.9 in' '75.7 in' '51.2 in' '71.4 in' '48.5 in'
 '63.5 in' '63 in' '58.2 in' '57.3 in' '69.6 in' '78.8 in' '65.6 in'
 '55 in' '56.6 in' '57.5 in' '67.8 in' '58.4 in' '73.7 in' '68 in'
 '63.6 in' '60.7 in' '62.9 in' '77.6 in' '63.2 in' '59.3 in' '59.4 in'
 '54.3 in' '59 in' '57 in' '62.7 in' '60 in' '70.3 in' '56.7 in' '77.7 in'
 '69.8 in' '61 in' '70.1 in' '69.1 in' '65.2 in' '55.5 in' '77.2 in'
 '64.4 in' '59.7 in' '72.7 in' '66.6 in' '54.8 in' '74.9 in' '78.5 in'
 '60.9 in' '55.6 in' '64.2 in' '64.7 in' '72.5 in' '68.2 in' '62.5 in'
 '61.5 in' '65.9 in' '75.8 in' '53.3 in' '62.6 in' '56 in' '61.6 in'
 '54 in' '76.7 in' '68.6 in' '65.1 in' '68.7 in' '54.5 in' '73.8 in'
```

```
 '72.2 in' '55.2 in' '59.2 in' '63.1 in' '67.2 in' '60.8 in' '76.6 in'
 '71.5 in' '70 in' '75.6 in' '77.4 in' '48.7 in' '77.5 in' '76.1 in'
 '78.4 in' '76.2 in' '68.1 in' '50.6 in' '65.4 in' '71.9 in' '76.4 in'
 '71.3 in' '66.4 in' '67.6 in' '56.4 in' '74.2 in' '56.9 in' '63.9 in'
 '75 in' '66.9 in' '78.1 in' '58.8 in' '73.4 in' '63.4 in' '75.9 in'
 '69.5 in' '61.8 in' '55.4 in' '71.6 in' '65.5 in' '72 in' '60.5 in'
 '60.1 in' '59.1 in' '48.8 in' '53.7 in' '62.2 in' '60.4 in' '56.1 in'
 '64.9 in' '71.7 in' '63.7 in' '53.5 in' '43.9 in' '108.6 in' '72.8 in'
 '70.5 in' '73.5 in' '74.6 in' '69.7 in' '59.5 in' '83.4 in' '75.3 in'
 '73 in' '78 in' '61.3 in' '72.3 in' '72.4 in' '61.7 in' '49.3 in'
 '46.5 in' '51.1 in' '45.9 in' '58.6 in' '50.9 in' '50.8 in' '51 in'
 '71.1 in' '50.7 in' '66.7 in' '54.1 in' '76 in' '75.1 in' '48.6 in'
 '53 in' '51.9 in' '49.1 in' '59.8 in' '51.4 in' '54.7 in' '50 in'
 '74.1 in' '82.4 in' '51.5 in' '76.5 in' '52.8 in' '65.8 in' '64.6 in'
 '71.8 in' '60.2 in' '52.1 in' '71.2 in' '50.4 in' '47.3 in' '47.6 in'
 '47.8 in' '68.8 in' '77 in' '53.9 in' '53.2 in' '63.8 in' '49.9 in'
 '75.2 in' '76.8 in' '78.3 in' '74.3 in' '62.4 in' '--' '99.2 in'
 '47.7 in' '45.8 in' '52.6 in' '52 in' '61.9 in' '59.9 in' '64 in'
 '107.7 in' '88 in' '51.3 in' '51.6 in' '50.1 in' '83.6 in' '77.3 in'
 '48.4 in' '60.6 in' '49 in' '51.7 in' '61.4 in' '84.1 in' '53.6 in'
 '46.6 in' '48.9 in' '74.8 in' '78.7 in' '52.2 in' '82.6 in' '47.1 in'
 '47.4 in' '47 in' '50.2 in' '50.3 in' '83.7 in' '46 in' '72.1 in'
 '52.7 in' '49.2 in' '84.6 in' '84.8 in' '46.3 in' '69.2 in' '79.3 in'
 '75.4 in' '53.8 in' '47.2 in' '52.3 in' '46.8 in' '48 in' '45.2 in'
 '44.3 in' '46.7 in' '49.4 in' '77.9 in' '80.2 in' '83.9 in' '82.2 in'
 '100.7 in' '46.4 in' '73.1 in' '74.5 in' '47.5 in' '49.7 in' '62.8 in'
 '73.2 in' '64.3 in' '50.5 in' '52.4 in' '49.8 in' '44 in' '82.1 in'
 '98.7 in' '52.5 in' '49.6 in' '44.7 in' '82.9 in' '101 in' '49.5 in'
 '77.8 in' '72.9 in' '82.3 in' '77.1 in' '74.7 in' '73.3 in' '100.8 in'
 '84.3 in' '81.2 in' '82.8 in' '48.1 in' '81.6 in' '78.2 in' '85 in'
 '80.9 in' '80.7 in' '84.5 in' '62 in' '48.3 in' '46.1 in' '83.2 in'
 '82.7 in' '48.2 in' '60.3 in' '53.4 in']
[True False]
[False  True]
['193.8 in' '184.8 in' '204.3 in' '203.7 in' '189.8 in' '175.1 in'
 '166.9 in' '181.1 in' '192.3 in' '166.6 in' '173.4 in' '201.2 in'
 '188.4 in' '200.8 in' '230 in' '183.3 in' '182.3 in' '192.5 in'
 '191.1 in' '198.3 in' '210 in' '189.9 in' '182.5 in' '192.6 in'
 '203.9 in' '176.2 in' '174.7 in' '203.6 in' '188.3 in' '167.6 in'
 '183.9 in' '184.9 in' '192.8 in' '182 in' '174.5 in' '173 in' '224.9 in'
 '198.5 in' '198.4 in' '224 in' '191.4 in' '187.4 in' '190.3 in'
 '231.7 in' '201.3 in' '186.3 in' '179.4 in' '202.8 in' '191.9 in'
 '203.8 in' '199.4 in' '179.9 in' '179.3 in' '184.5 in' '172.4 in'
 '180.9 in' '173.8 in' '184.1 in' '200 in' '192.9 in' '212.4 in'
 '194.5 in' '212.3 in' '185.1 in' '189.5 in' '190.6 in' '179.5 in'
 '224.3 in' '196.2 in' '189.6 in' '182.8 in' '176.9 in' '212.7 in'
 '185.5 in' '205 in' '190.9 in' '195.5 in' '188.8 in' '196.1 in'
 '191.3 in' '160.1 in' '183.5 in' '178.1 in' '177.3 in' '201.4 in'
```

'202.9 in' '191.5 in' '181.8 in' '192.1 in' '209.3 in' '229 in'
'162.8 in' '202.5 in' '159.6 in' '163 in' '165.2 in' '194.6 in'
'178.7 in' '183.4 in' '197.2 in' '182.6 in' '185.7 in' '185 in'
'182.2 in' '168 in' '182.7 in' '197.9 in' '175.4 in' '190.4 in'
'190.7 in' '202 in' '229.9 in' '185.3 in' '189 in' '181 in' '175.8 in'
'193.1 in' '189.1 in' '232.9 in' '195.7 in' '229.8 in' '169.1 in'
'188.5 in' '187.2 in' '200.9 in' '186.8 in' '179.1 in' '190.2 in'
'179.2 in' '229.5 in' '184.4 in' '161.6 in' '222.9 in' '185.6 in'
'204 in' '183.6 in' '183.1 in' '182.1 in' '193.9 in' '171.9 in'
'172.6 in' '180.7 in' '185.9 in' '199.6 in' '176.4 in' '231.9 in'
'198.6 in' '174.8 in' '189.2 in' '180.2 in' '200.2 in' '205.8 in'
'180 in' '197.1 in' '198 in' '180.3 in' '168.3 in' '181.9 in' '175.6 in'
'181.5 in' '178.3 in' '190 in' '191.8 in' '199.8 in' '180.6 in'
'168.4 in' '188.9 in' '202.2 in' '187.8 in' '173.2 in' '196.5 in'
'172.8 in' '200.6 in' '191.6 in' '177.8 in' '184.6 in' '204.9 in'
'208.3 in' '185.2 in' '173.6 in' '192.2 in' '177.9 in' '183.8 in'
'162 in' '168.7 in' '193.5 in' '189.7 in' '194.2 in' '164 in' '203.2 in'
'182.9 in' '202.4 in' '170.9 in' '188.1 in' '175.5 in' '187.1 in'
'224.4 in' '174.3 in' '186.7 in' '184.2 in' '193.3 in' '159.3 in'
'152.8 in' '167 in' '205.6 in' '195 in' '172 in' '191.7 in' '190.5 in'
'175.9 in' '205.5 in' '204.1 in' '201.8 in' '176.5 in' '221.3 in'
'197.5 in' '194.9 in' '215 in' '200.1 in' '192.4 in' '237.9 in'
'199.9 in' '193.6 in' '241.8 in' '218 in' '221.9 in' '194.1 in'
'230.2 in' '166.7 in' '194.7 in' '202.1 in' '200.3 in' '191 in'
'196.8 in' '193.4 in' '183.7 in' '208.9 in' '184.3 in' '164.3 in'
'198.8 in' '181.3 in' '167.2 in' '161.8 in' '186.1 in' '171.2 in'
'195.6 in' '169.3 in' '171.3 in' '195.3 in' '170.5 in' '228.9 in'
'196.9 in' '207.4 in' '206.5 in' '179 in' '176 in' '193.2 in' '206.6 in'
'183 in' '178.8 in' '171.7 in' '197.7 in' '162.4 in' '146.2 in'
'194.3 in' '205.3 in' '161.3 in' '171.6 in' '200.4 in' '157.4 in'
'228.7 in' '172.2 in' '192.7 in' '196.6 in' '167.5 in' '160 in'
'163.7 in' '154.1 in' '164.9 in' '195.4 in' '174.2 in' '192 in'
'201.9 in' '170.4 in' '161.7 in' '168.2 in' '196.4 in' '243.7 in'
'229.4 in' '155.5 in' '149 in' '235.5 in' '201.6 in' '225.5 in'
'200.7 in' '208.1 in' '207.2 in' '193.7 in' '224.6 in' '151.9 in'
'190.1 in' '188.2 in' '186.2 in' '180.5 in' '171.5 in' '216.7 in'
'166.8 in' '177 in' '222.3 in' '169.8 in' '178.2 in' '161.4 in'
'208.5 in' '220.9 in' '190.8 in' '202.6 in' '176.8 in' '212.6 in'
'154.7 in' '208.6 in' '177.5 in' '194 in' '207.9 in' '220.8 in'
'203.3 in' '219.3 in' '177.1 in' '178.5 in' '197 in' '180.1 in'
'194.8 in' '173.9 in' '194.4 in' '184.7 in' '206.9 in' '191.2 in'
'228.1 in' '207.6 in' '168.8 in' '195.1 in' '169.6 in' '174.6 in'
'172.1 in' '144.4 in' '169 in' '198.9 in' '160.7 in' '188 in' '237.6 in'
'195.9 in' '201.7 in' '207.1 in' '209 in' '165 in' '187 in' '196.7 in'
'176.3 in' '187.5 in' '207 in' '206 in' '159 in' '207.7 in' '178.6 in'
'195.2 in' '205.1 in' '175 in' '229.3 in' '150.9 in' '179.6 in'
'162.2 in' '186 in' '210.2 in' '158 in' '176.6 in' '164.7 in' '186.5 in'
'208.7 in' '197.3 in' '182.4 in' '204.4 in' '145.6 in' '159.7 in'

```
 '204.7 in' '206.1 in' '184 in' '222.4 in' '239.6 in' '212 in' '152.7 in'
 '227.7 in' '169.5 in' '201 in' '223.8 in' '--' '176.7 in' '179.7 in'
 '173.5 in' '189.3 in' '203 in' '187.6 in' '173.1 in' '181.7 in'
 '170.7 in' '231.8 in' '186.4 in' '187.9 in' '151.1 in' '219.5 in'
 '169.9 in' '139.6 in' '193 in' '149.4 in' '263.9 in' '187.3 in'
 '196.3 in' '150.6 in' '180.4 in' '228.5 in' '185.8 in' '186.6 in'
 '219.9 in' '211.2 in' '172.3 in' '201.5 in' '207.8 in' '183.2 in'
 '171 in' '225.9 in' '166.1 in' '177.4 in' '227.9 in' '250.5 in'
 '161.5 in' '213.2 in' '165.9 in' '197.6 in' '197.4 in' '175.2 in'
 '167.8 in' '171.1 in' '181.6 in' '206.2 in' '188.7 in' '163.3 in'
 '244.1 in' '179.8 in' '165.3 in' '150.1 in' '220.6 in' '227.5 in'
 '242.8 in' '168.9 in' '231 in' '177.2 in' '209.1 in' '174.4 in'
 '177.6 in' '148.8 in' '206.7 in' '167.3 in' '211.4 in' '203.1 in'
 '156.7 in' '199.1 in' '180.8 in' '147 in' '219.4 in' '155.3 in'
 '224.1 in' '235.8 in' '165.8 in' '185.4 in' '178.4 in' '214.7 in'
 '178.9 in' '165.4 in' '171.4 in' '159.4 in' '144.7 in' '218.3 in'
 '155 in' '157 in' '250.3 in' '157.2 in' '168.1 in' '159.8 in' '221.6 in'
 '174.1 in' '211.1 in' '215.4 in' '163.1 in' '206.8 in' '174.9 in'
 '169.7 in' '201.1 in' '164.5 in' '155.4 in' '166.5 in' '200.5 in'
 '232.1 in' '176.1 in' '166 in' '236.7 in' '247.8 in' '168.5 in' '174 in'
 '209.8 in' '170.3 in' '161.1 in' '106.1 in' '163.2 in' '169.4 in'
 '167.7 in' '163.8 in' '199.2 in' '189.4 in' '211.9 in' '154.3 in'
 '160.5 in' '243.9 in' '226.2 in' '164.8 in' '225.8 in' '172.5 in'
 '208.4 in' '199.3 in' '198.1 in' '167.4 in' '167.9 in' '249.2 in'
 '218.5 in' '205.9 in' '230.1 in' '153.5 in' '163.5 in' '159.1 in'
 '213.1 in' '215.1 in' '178 in' '217.5 in' '175.7 in' '196 in' '207.3 in'
 '195.8 in' '215.3 in' '198.2 in' '218.8 in' '166.2 in' '158.5 in'
 '146.8 in' '229.6 in' '239 in' '217.8 in' '154.9 in' '224.2 in'
 '205.7 in' '152.5 in' '172.9 in' '143.1 in' '155.8 in' '172.7 in'
 '247.6 in' '222 in' '175.3 in' '236 in' '158.1 in' '224.5 in' '266.1 in'
 '162.9 in' '250.4 in' '188.6 in' '143.9 in' '202.7 in' '225.1 in'
 '152 in' '223.3 in' '220.1 in' '157.3 in' '202.3 in' '247.9 in'
 '237.2 in' '157.1 in' '148 in' '177.7 in' '164.4 in' '186.9 in' '199 in'
 '170.2 in' '156.1 in' '165.1 in' '221.4 in' '157.9 in' '218.9 in'
 '155.7 in' '206.3 in' '211.8 in' '243.2 in' '199.7 in' '157.5 in'
 '220.2 in' '173.7 in' '181.2 in' '153 in' '211.5 in' '222.1 in'
 '241.2 in' '246.6 in' '210.7 in' '228.2 in' '214.8 in' '237.3 in'
 '221.7 in' '151.8 in' '236.4 in' '170.1 in' '155.9 in' '214.1 in'
 '249 in' '197.8 in' '160.4 in' '227.6 in' '161.9 in' '246.7 in'
 '229.7 in' '204.8 in' '220.3 in' '239.7 in' '209.7 in' '241.3 in'
 '215.8 in' '142.8 in' '164.6 in' '152.6 in' '205.4 in' '238.9 in'
 '208 in' '147.2 in' '205.2 in' '226.9 in' '227.2 in' '147.7 in'
 '212.2 in' '207.5 in']
['SILVER' 'BLACK' 'RED' 'WHITE' 'UNKNOWN' 'BLUE' 'GRAY' 'BROWN' 'YELLOW'
 'ORANGE' 'GREEN' 'PURPLE' 'TEAL' 'GOLD' 'PINK']
['Chevrolet' 'Lexus' 'Jeep' 'Hyundai' 'Cadillac' 'Chrysler' 'Dodge'
 'Nissan' 'Honda' 'Mercedes-Benz' 'Kia' 'Ford' 'Lincoln' 'Subaru' 'BMW'
 'Audi' 'Volkswagen' 'Jaguar' 'Mazda' 'GMC' 'Toyota' 'Acura' 'INFINITI'
```

```
 'RAM' 'FIAT' 'Volvo' 'Buick' 'Land Rover' 'Mitsubishi' 'Genesis'
 'Maserati' 'Saab' 'Bentley' 'MINI' 'Alfa Romeo' 'Porsche' 'Lotus'
 'Rolls-Royce' 'Lamborghini' 'Scion' 'Saturn' 'Ferrari' 'Pontiac'
 'Mercury' 'Suzuki' 'Aston Martin' 'Oldsmobile' 'McLaren' 'smart' 'Isuzu'
 'Plymouth' 'Hummer' 'SRT' 'Bugatti']
['5 seats' '4 seats' '8 seats' '7 seats' '6 seats' '2 seats' '3 seats'
 '15 seats' '12 seats' '9 seats' '--' '10 seats']
['Malibu' 'RC 350' 'Traverse' 'Grand Cherokee' 'Compass' 'Veloster' 'XT4'
 '200' 'Renegade' 'Wrangler Unlimited' 'Durango' 'Charger'
 'Silverado 1500' 'Rogue' 'Civic' 'RX 350' 'GLC-Class' 'Optima' 'Explorer'
 'Navigator' 'Outback' '4 Series' 'GLE-Class' 'Escalade' 'Tucson'
 '2 Series' 'Pacifica' 'Camaro' 'Trax' 'SQ5' 'Maxima' 'Cherokee' 'Tiguan'
 'Colorado' 'Pathfinder' 'Suburban' 'Blazer' 'Sorento' 'SRX' 'Mustang'
 'Impala' 'F-PACE' 'Town & Country' 'Altima' 'CX-9' 'Elantra' 'CX-5'
 'C-Class' 'Rogue Sport' 'WRX' 'Sonata' 'Grand Caravan' 'Patriot'
 'Santa Fe' 'LS 460' 'Accord' 'Canyon' 'Pilot' 'Passat' 'Tacoma' 'A4'
 'XT5' 'Forte' 'Escalade ESV' 'MDX' 'ATS' 'Liberty' 'Terrain' 'S-Class'
 'Camry' 'CTS' 'Edge' 'Acadia' 'Q70' 'Fiesta' 'RAV4' 'Escape' 'Sedona'
 'Odyssey' 'MAZDA6' 'SL-Class' 'E-Class' 'F-150' '1500' 'SLK-Class'
 '124 Spider' 'RAM 1500' 'Soul' '5 Series' 'X3' 'CTS-V' 'Santa Fe Sport'
 'Q60' 'Jetta' 'GTI' 'Challenger' 'A3' 'Arteon' 'Tahoe' 'V60' 'Cruze'
 'Murano' 'Crosstrek' 'Taurus' 'X1' 'M-Class' 'Cadenza' 'Sierra 1500'
 'HR-V' 'Legacy' 'XTS' 'RDX' 'Regal' 'Corolla' 'MKC' 'TLX' 'XT6' 'QX50'
 'Fit' 'WRX STI' 'TSX' 'Stinger' 'Impreza' 'MKZ' 'Matrix' 'Highlander'
 'Rio' 'Discovery Sport' 'Q5' 'Q7' 'Sportage' '300' 'Golf Alltrack'
 'Sienna' 'Routan' 'Palisade' 'CT5' 'CT6' 'S4' 'Xterra' 'MAZDA3' 'CX-3'
 'ILX' 'S60' 'MAZDA5' 'CR-V' 'Trailblazer' 'S5' 'Sentra' 'Golf R' 'CC'
 'A8' 'Forester' 'Yukon' 'Outlander' 'A7' 'G80' 'Q3' 'XC70' 'SC 430'
 '3 Series' 'Focus RS' 'G90' 'QX80' 'A4 Avant' 'GranTurismo'
 'Civic Hatchback' 'Equinox' 'Accent' 'A6' 'Kona' 'A5' 'Focus' '9-3'
 'Fusion' 'MKX' 'Bentayga' 'Elantra GT' 'Sonata Hybrid' 'S5 Sportback'
 'ES 350' 'Wrangler' 'XC60' 'XC90' 'Passport' 'RX' 'Frontier'
 'Continental' 'Golf' 'Flex' 'Avalanche' '4Runner' 'Dakota' 'S90'
 'Journey' 'Corvette' 'Q50' 'Gladiator' 'Expedition' 'BRZ'
 '3 Series Gran Turismo' 'M37' 'Genesis' 'Range Rover' 'NX 200t' 'IS'
 'Armada' 'GX' 'XF' 'Countryman' 'C-HR' 'X4' 'Forte5' 'Avalon'
 'Corolla iM' 'Telluride' 'QX60' 'QX56' 'X5' 'Civic Coupe' 'Venza'
 '7 Series' 'Tundra' 'MKS' 'Yaris iA' 'Juke' 'Cooper' 'X6' 'Ascent'
 'EcoSport' 'NX' 'CLA-Class' 'GS 350' 'Encore' 'Enclave' 'FJ Cruiser'
 'Versa Note' 'Stelvio' 'X2' 'X5 M' 'Q8' 'MX-5 Miata' '6 Series' 'CX-30'
 'G37' 'XC40' 'Range Rover Sport' 'Cayenne' 'Versa' 'Quest' 'Yukon XL'
 'Accord Coupe' 'IS 250' '500X' 'Nautilus' 'JX35' 'Outlander Sport'
 'Giulia' 'Yaris' 'MAZDA2' 'Voyager' 'Continental GTC' 'Elise' 'Dart'
 'Transit Passenger' 'GLS-Class' 'Corolla Hatchback' 'Lancer'
 'Quattroporte' 'Accord Crosstour' 'LaCrosse' 'Azera' 'Discovery' 'Titan'
 'Genesis Coupe' 'M4' 'Murano CrossCabriolet' 'M3' 'Ridgeline' 'M5' 'M2'
 'GX 470' 'Cooper Clubman' 'RX 330' 'Mustang SVT Cobra' 'Cruze Limited'
 'E-Series' 'S-Class Coupe' 'G-Class' 'Corsair' 'GLK-Class'
```

'Cayenne E-Hybrid' 'Dawn' 'GL-Class' 'Macan' 'Phantom Drophead Coupe'
'Panamera' 'SLR McLaren' 'Huracan' '911' 'Crosstour' 'Urus' 'Ghost' 'xD'
'Flying Spur' 'Continental GT' 'TL' 'Wraith' 'Phantom' 'X7' 'Ghibli'
'Levante' 'G70' 'Rogue Select' 'GLA-Class' 'V90' '370Z' 'NV200'
'CL-Class' 'MKT' '9-5' 'Impreza WRX STI' 'Verano' 'FR-S' 'CTS Coupe'
'Beetle' 'Range Rover Velar' 'RL' 'S-Series' '350Z' 'Range Rover Evoque'
'500' 'E-PACE' 'Lancer Evolution' 'ES' 'LX 570' 'ES 330' 'XJ-Series'
'Metris' 'F-TYPE' 'DTS' 'TTS' 'CLS-Class' 'Camry Solara' 'QX30' 'FX35'
'Q70L' 'Thunderbird' 'Sonic' 'Sebring' '718 Cayman' '458 Italia'
'Mustang Shelby GT500' 'GTO' 'Viper' 'G8' 'Sequoia' 'HHR'
'Golf SportWagen' 'S2000' 'Jetta GLI' 'RS 5' 'Explorer Sport Trac' 'S40'
'ATS Coupe' 'Milan' 'A4 Allroad' 'TT' 'S3' 'ProMaster City' 'RS 7'
'GLB-Class' 'RX 300' 'A5 Sportback' 'Cayman' 'LS 500' 'Cobalt' 'SX4'
'CX-7' 'Crown Victoria' 'Entourage' 'Aveo' 'Envision' 'Jetta SportWagen'
'Optima Hybrid' 'Kicks' 'Mirage G4' 'Civic Type R' 'IS 350' 'R-Class'
'Impala Limited' 'RC 300' 'ES 300' '718 Boxster' 'Z4' '86' 'EX35'
'Mulsanne' 'Element' 'RS 3' 'Mirage' 'X4 M' 'XE' 'SLC-Class' 'G35'
'California T' 'SSR' 'Vanquish' 'DB9' 'DBS' 'DB11' 'Vantage' 'Vibe'
'Transit Cargo' 'Cavalier' 'Windstar' 'X3 M' 'Trailblazer EXT'
'Santa Fe XL' 'Caliber' 'Accord Hybrid' 'iA' 'GS' 'Boxster' 'Venue'
'Sable' 'Altima Coupe' 'RC' 'Evora' 'ATS-V Coupe' '599 GTB Fiorano'
'Metris Cargo' 'F12 Berlinetta' 'A-Class' 'CTS-V Coupe' 'R8' 'S8' 'G25'
'Touareg' 'Civic Hybrid' 'Impreza WRX' 'XV Crosstrek Hybrid' 'Carrera GT'
'LR2' 'Outlook' 'Continental Flying Spur' 'Rondo' 'ECHO' 'Express'
'Mountaineer' 'Envoy' 'Eldorado' 'Continental Supersports' 'LeSabre'
'Rendezvous' 'Elantra Touring' '9-3 SportCombi' 'RC F' 'Gallardo'
'8 Series' 'LC' 'S80' 'iM' 'Cooper Paceman' 'Avenger' 'Ranger'
'PT Cruiser' 'Mustang Shelby GT350' 'Concorde' 'Allante' 'Torrent'
'Aztek' 'Five Hundred' 'Captiva Sport' 'Aura' 'X6 M' 'M6' 'Q40'
'Veloster Turbo' 'Rio5' 'Alero' '5 Series Gran Turismo' 'Veracruz'
'Equus' 'Cube' 'VUE' 'Lucerne' 'C70' 'ATS-V' 'Roadster' 'Express Cargo'
'Rabbit' 'Cooper Coupe' 'M8' 'F430 Spider' '600LT' '650S' '570S'
'RS 5 Sportback' '720S' '812 Superfast' 'Land Cruiser' 'Superamerica'
'612 Scaglietti' 'TT RS' 'Forte Koup' 'Encore GX' '1 Series'
'XF Sportbrake' 'Z3' 'Spark' '4C' '500L' 'CTS-V Wagon' 'LR4'
'Sierra 1500 Limited' 'Solstice' 'Commander' 'Firebird' 'Eclipse'
'Monte Carlo' 'Crossfire' 'Nitro' 'L300' 'DeVille' 'Eos' 'Grand Marquis'
'Town Car' 'Monterey' 'ION' 'Cullinan' 'A6 Allroad' 'STS'
'Regal Sportback' 'tC' 'CLK-Class' 'RC 200t' 'Regal TourX' 'K900'
'Mariner' 'City Express' 'Transit Connect' 'Classic' 'Eclipse Spyder'
'XK-Series' 'Seltos' '488' '570GT' 'GTC4Lusso' 'Sky' 'fortwo' 'xB'
'Rapide' 'Grand Wagoneer' '675LT' 'Tribeca' 'LFA' 'Enzo' 'FF' 'GT'
'Envoy XL' 'AMG GT' 'Grand Prix' 'Tribute' 'Aviator' 'UX' 'Regency'
'V8 Vantage' 'Cutlass Supreme' 'Crosstrek Hybrid' 'Defender' 'SS' 'GT-R'
'Freestyle' 'Galant' 'Savana' 'G6' 'Kizashi' 'Grand Vitara'
'Range Rover Hybrid' 'S-TYPE' 'M30' '3000GT' 'Astra' 'X-TYPE' '9-7X'
'C30' 'LS 430' 'Rodeo' 'LS' 'Veloster N' 'Silverado Classic 1500'
'Savana Cargo' 'S6' 'Bonneville' 'Freestar' 'Rainier' 'Marauder'

```
 'Prowler' 'B9 Tribeca' 'Montego' 'MAZDASPEED6' 'QX4' 'Portofino'
 'SLS-Class' 'California' '360 Spider' 'Prizm' 'XL-7' 'Spectra' '550'
 'GTC4Lusso T' 'Amanti' 'Century' 'ZDX' 'V70' 'MAZDASPEED3'
 'Panamera E-Hybrid' 'Endeavor' 'Lumina' 'Uplander' 'H3'
 '6 Series Gran Turismo' '626' 'Tiburon' 'Park Avenue' 'Aspen' 'Magnum'
 'Escape Hybrid' 'Z3 M' 'M35' 'Jimmy' 'Sierra Classic 1500' 'LX' 'GS 200t'
 'Elantra Coupe' 'Stealth' 'Malibu Maxx' 'Caravan' 'Aventador'
 '430 Scuderia' 'Panamera Hybrid' 'Mark VIII' 'Escalade EXT' 'H3T' 'R32'
 'FX50' 'ProMaster' 'CL' '944' 'ILX Hybrid' 'Explorer Sport' 'Celica'
 'F430' 'GS 430' 'Murcielago' 'Riviera' 'Intrepid' 'Envoy XUV' 'Fleetwood'
 'S-TYPE R' 'Stratus' 'Forenza' 'Verona' 'Equator' 'Taurus X' 'Baja'
 'Lancer Sportback' '968' 'Protege5' 'S-10' 'G3' 'Aurora' 'Villager'
 'Seville' 'V40' 'Silverado 1500HD' 'V50' 'GS 300' 'C/V' 'Supra'
 'RLX Hybrid Sport' 'Montana SV6' 'Raider' 'Neon' 'M56' 'Prelude' 'CR-Z'
 'Intrigue' 'Astro Cargo' 'Montana' 'Veyron' 'F-150 Heritage' 'Silhouette'
 'Venture' 'Diamante' 'Eighty-Eight' 'RSX' 'Astro' 'Contour' 'Zephyr'
 'Grand Am' 'Silverado SS' 'Blackwood' 'Sunfire' 'M45'
 'Range Rover Hybrid Plug-in' 'Touareg 2' 'RX-8' 'Escort'
 'CTS Sport Wagon' 'LR3' 'Neon SRT-4' 'LX 470' 'Crossfire SRT-6' 'XLR-V'
 'ActiveHybrid 3' 'Montero' 'L-Series' 'Fiero' 'Mark LT' 'Borrego'
 'B-Series' 'CT4' 'Sierra 2500HD' 'MR2 Spyder' 'Cayenne Hybrid' 'IPL G'
 'Sonoma' 'GS F' '9-4X' 'Vitara' 'XLR' 'Safari' '575M' 'Q70 Hybrid'
 'Cirrus' 'LS 400' 'EuroVan' 'Bravada' '300M' 'Cabrio' 'Tracker' 'MPV'
 'Z4 M' 'Jetta Hybrid' 'Cougar' 'MP4-12C' '300ZX' '1M' '380-Class' 'XG350'
 'New Yorker' 'S7' 'i-Series' 'Silverado Classic 1500HD' 'G5'
 'ActiveHybrid 7' 'C/K 1500' 'Windstar Cargo' 'Skylark' 'Electra' 'I30'
 'Metro' 'Montero Sport' 'Terraza' 'Roadmaster' 'NSX' 'Mark VII' 'LHS'
 'Mariner Hybrid' 'Truck' 'Trooper' 'xA' 'F-150 SVT Lightning'
 '918 Spyder' 'V12 Vanquish' 'Reno' '300-Class' 'Eighty-Eight Royale'
 'Civic del Sol' '360' 'Sierra 1500HD' 'Integra' 'Arnage' 'Allroad'
 'V70 R' 'Z8' 'V12 Vantage' '960' 'GS Hybrid']
['160 hp @ 5,700 RPM' '311 hp @ 6,600 RPM' '310 hp @ 6,800 RPM' …
 '420 hp @ 7,000 RPM' '340 hp @ 6,400 RPM' '168 hp @ 5,500 RPM']
[False True]
['184 lb-ft @ 2,500 RPM' '280 lb-ft @ 4,800 RPM' '266 lb-ft @ 2,800 RPM'
 … '267 lb-ft @ 4,800 RPM' '220 lb-ft @ 3,800 RPM'
 '177 lb-ft @ 4,500 RPM']
['A' 'M' 'CVT' 'Dual Clutch']
['FWD' 'AWD' '4WD' 'RWD' '4X2']
['73 in' '81.5 in' '78.6 in' '78.5 in' '84.8 in' '71.4 in' '70.9 in'
 '83.5 in' '73.6 in' '79.6 in' '82.8 in' '85.5 in' '73.8 in' '75 in'
 '80 in' '70.8 in' '74.6 in' '82.5 in' '73.2 in' '90.2 in' '93.8 in'
 '81.3 in' '71.8 in' '84.3 in' '80.5 in' '72.8 in' '78.1 in' '90.4 in'
 '72.4 in' '74.7 in' '69.9 in' '73.7 in' '71.2 in' '83.9 in' '77.2 in'
 '76.7 in' '74.4 in' '75.2 in' '81.9 in' '81.2 in' '85.6 in' '69 in'
 '88.5 in' '72 in' '77.5 in' '79.5 in' '72.3 in' '78.9 in' '73.4 in'
 '79.3 in' '69.2 in' '72.9 in' '72.2 in' '80.3 in' '75.5 in' '70.1 in'
 '77.7 in' '71.1 in' '71.7 in' '80.8 in' '85.8 in' '72.6 in' '72.5 in'
```

```
 '67.8 in' '81.8 in' '89.3 in' '91.5 in' '82.6 in' '96.8 in' '79.4 in'
 '79 in' '68.5 in' '83.7 in' '87.5 in' '74 in' '70 in' '70.5 in' '74.9 in'
 '81.1 in' '79.9 in' '70.7 in' '74.1 in' '85.7 in' '82.1 in' '69.8 in'
 '80.1 in' '81 in' '74.8 in' '73.1 in' '84.1 in' '78.4 in' '77.3 in'
 '71 in' '66.7 in' '79.1 in' '82.2 in' '83.3 in' '69.5 in' '75.8 in'
 '67.9 in' '84.2 in' '87.1 in' '77.8 in' '69.6 in' '70.6 in' '69.1 in'
 '74.2 in' '71.6 in' '76.2 in' '69.3 in' '83.1 in' '71.3 in' '71.5 in'
 '75.4 in' '76 in' '80.9 in' '66.9 in' '92.3 in' '78 in' '97 in' '86.1 in'
 '79.2 in' '82.3 in' '73.9 in' '88.8 in' '83.4 in' '85.4 in' '75.9 in'
 '93.4 in' '73.3 in' '80.6 in' '77.1 in' '87.4 in' '79.8 in' '78.8 in'
 '78.3 in' '74.5 in' '84.9 in' '91.8 in' '70.4 in' '86 in' '86.5 in'
 '81.4 in' '75.7 in' '87.3 in' '84 in' '66.2 in' '82 in' '86.2 in' '67 in'
 '68.3 in' '69.4 in' '85.2 in' '73.5 in' '77.6 in' '86.7 in' '79.7 in'
 '69.7 in' '68.9 in' '72.7 in' '87.7 in' '97.4 in' '82.7 in' '78.2 in'
 '76.1 in' '80.2 in' '68.1 in' '95.7 in' '82.9 in' '75.6 in' '83 in'
 '82.4 in' '86.4 in' '75.1 in' '88 in' '76.3 in' '77.9 in' '85.9 in'
 '86.9 in' '83.2 in' '85 in' '71.9 in' '84.4 in' '68.2 in' '88.3 in'
 '80.4 in' '68.6 in' '70.3 in' '72.1 in' '78.7 in' '81.7 in' '77.4 in'
 '76.8 in' '68.7 in' '67.6 in' '84.6 in' '74.3 in' '70.2 in' '65.7 in'
 '--' '67.5 in' '68 in' '85.1 in' '65.6 in' '80.7 in' '98.6 in' '76.6 in'
 '68.8 in' '76.5 in' '75.3 in' '76.4 in' '77 in' '65.4 in' '67.1 in'
 '66 in' '62.9 in' '67.3 in' '83.8 in' '67.2 in' '86.3 in' '76.9 in'
 '61.4 in' '87.6 in' '67.4 in' '66.1 in' '66.5 in' '89.9 in' '66.3 in'
 '84.5 in' '64.1 in' '87.2 in' '65.9 in' '66.4 in' '65 in' '89.1 in'
 '98 in' '67.7 in' '66.6 in' '89.2 in' '89.5 in' '64 in' '68.4 in'
 '66.8 in' '65.3 in' '63.7 in' '81.6 in' '62.6 in']
```

finding all unique values per column to see what values we need to clean

[24]:
```
columns_that_need_cleaning_2 =␣
 ↪['fuel_tank_volume','height','length','maximum_seating','width',]
for col in columns_that_need_cleaning_2:
        print(df_cln_2[col].unique())
```

```
['15.8 gal' '17.4 gal' '19.4 gal' '22 gal' '24.6 gal' '13.5 gal'
 '13.2 gal' '15.9 gal' '12.7 gal' '22.5 gal' '21.5 gal' '18.5 gal'
 '26 gal' '12.4 gal' '19.2 gal' '18.6 gal' '23.2 gal' '24 gal' '16.4 gal'
 '13.7 gal' '19 gal' '14.5 gal' '14 gal' '18 gal' '16.9 gal' '21 gal'
 '19.5 gal' '31 gal' '18.8 gal' '16 gal' '16.6 gal' '21.7 gal' '20 gal'
 '15.3 gal' '22.2 gal' '17.2 gal' '21.1 gal' '16.1 gal' '13 gal'
 '23.8 gal' '17 gal' '15.1 gal' '19.8 gal' '23 gal' '20.5 gal' '11.9 gal'
 '14.2 gal' '14.3 gal' '14.8 gal' '17.7 gal' '16.5 gal' '17.8 gal'
 '15.6 gal' '17.1 gal' '15.5 gal' '10.6 gal' '15.7 gal' '18.1 gal'
 '20.1 gal' '12.8 gal' '20.3 gal' '21.6 gal' '21.9 gal' '11.4 gal'
 '36 gal' '16.8 gal' '15.4 gal' '16.3 gal' '28.3 gal' '17.9 gal'
 '27.7 gal' '13.6 gal' '17.5 gal' '28 gal' '27 gal' '22.4 gal' '20.6 gal'
 '26.4 gal' '11.6 gal' '11.8 gal' '12 gal' '19.3 gal' '13.1 gal'
 '10.8 gal' '14.9 gal' '19.1 gal' '12.1 gal' '27.3 gal' '38 gal'
```

```
 '11.3 gal' '32 gal' '10 gal' '23.3 gal' '25 gal' '11.1 gal' '16.2 gal'
 '23.5 gal' '15 gal' '33 gal' '23.6 gal' '25.4 gal' '33.5 gal' '17.6 gal'
 '23.7 gal' '25.6 gal' '21.8 gal' '18.4 gal' '10.5 gal' '14.6 gal'
 '20.4 gal' '14.4 gal' '27.6 gal' '12.2 gal' '22.7 gal' '25.1 gal'
 '18.3 gal' '12.6 gal' '18.2 gal' '11 gal' '17.3 gal' '9.2 gal' '30 gal'
 '--' '24.3 gal' '27.8 gal' '28.5 gal' '22.8 gal' '31.5 gal' '34 gal'
 '8.7 gal' '23.9 gal' '29 gal' '18.7 gal' '20.2 gal' '35 gal' '15.2 gal'
 '21.4 gal' '16.7 gal' '39 gal' '40.7 gal' '21.2 gal' '8 gal' '12.3 gal'
 '24.1 gal' '42 gal' '13.3 gal' '7.7 gal' '7.8 gal']
['57.6 in' '55.1 in' '70.7 in' '69.9 in' '69.3 in' '65 in' '64.1 in'
 '58.7 in' '66.5 in' '70.8 in' '70.9 in' '73.6 in' '58.5 in' '73.9 in'
 '65.3 in' '55.7 in' '67.7 in' '64.5 in' '58.3 in' '57.7 in' '71 in'
 '76.3 in' '66.1 in' '54.6 in' '74.4 in' '64.8 in' '55.8 in' '69.4 in'
 '66.3 in' '53.1 in' '66 in' '72.6 in' '56.5 in' '66.2 in' '67.1 in'
 '65.7 in' '70.6 in' '70.2 in' '67 in' '54.4 in' '75.5 in' '58.9 in'
 '74 in' '67.9 in' '57.8 in' '69 in' '52.9 in' '67.3 in' '56.8 in'
 '63.3 in' '58.1 in' '68.9 in' '55.3 in' '66.8 in' '57.4 in' '70.4 in'
 '56.2 in' '54.2 in' '56.3 in' '67.4 in' '55.9 in' '58 in' '57.9 in'
 '57.2 in' '68.3 in' '59.6 in' '54.9 in' '68.5 in' '67.5 in' '68.4 in'
 '57.1 in' '51.8 in' '76.9 in' '75.7 in' '51.2 in' '71.4 in' '48.5 in'
 '63.5 in' '63 in' '58.2 in' '57.3 in' '69.6 in' '78.8 in' '65.6 in'
 '55 in' '56.6 in' '57.5 in' '67.8 in' '58.4 in' '73.7 in' '68 in'
 '63.6 in' '60.7 in' '62.9 in' '77.6 in' '63.2 in' '59.3 in' '59.4 in'
 '54.3 in' '59 in' '57 in' '62.7 in' '60 in' '70.3 in' '56.7 in' '77.7 in'
 '69.8 in' '61 in' '70.1 in' '69.1 in' '65.2 in' '55.5 in' '77.2 in'
 '64.4 in' '59.7 in' '72.7 in' '66.6 in' '54.8 in' '74.9 in' '78.5 in'
 '60.9 in' '55.6 in' '64.2 in' '64.7 in' '72.5 in' '68.2 in' '62.5 in'
 '61.5 in' '65.9 in' '75.8 in' '53.3 in' '62.6 in' '56 in' '61.6 in'
 '54 in' '76.7 in' '68.6 in' '65.1 in' '68.7 in' '54.5 in' '73.8 in'
 '72.2 in' '55.2 in' '59.2 in' '63.1 in' '67.2 in' '60.8 in' '76.6 in'
 '71.5 in' '70 in' '75.6 in' '77.4 in' '48.7 in' '77.5 in' '76.1 in'
 '78.4 in' '76.2 in' '68.1 in' '50.6 in' '65.4 in' '71.9 in' '76.4 in'
 '71.3 in' '66.4 in' '67.6 in' '56.4 in' '74.2 in' '56.9 in' '63.9 in'
 '75 in' '66.9 in' '78.1 in' '58.8 in' '73.4 in' '63.4 in' '75.9 in'
 '69.5 in' '61.8 in' '55.4 in' '71.6 in' '65.5 in' '72 in' '60.5 in'
 '60.1 in' '59.1 in' '48.8 in' '53.7 in' '62.2 in' '60.4 in' '56.1 in'
 '64.9 in' '71.7 in' '63.7 in' '53.5 in' '43.9 in' '108.6 in' '72.8 in'
 '70.5 in' '73.5 in' '74.6 in' '69.7 in' '59.5 in' '83.4 in' '75.3 in'
 '73 in' '78 in' '61.3 in' '72.3 in' '72.4 in' '61.7 in' '49.3 in'
 '46.5 in' '51.1 in' '45.9 in' '58.6 in' '50.9 in' '50.8 in' '51 in'
 '71.1 in' '50.7 in' '66.7 in' '54.1 in' '76 in' '75.1 in' '48.6 in'
 '53 in' '51.9 in' '49.1 in' '59.8 in' '51.4 in' '54.7 in' '50 in'
 '74.1 in' '82.4 in' '51.5 in' '76.5 in' '52.8 in' '65.8 in' '64.6 in'
 '71.8 in' '60.2 in' '52.1 in' '71.2 in' '50.4 in' '47.3 in' '47.6 in'
 '47.8 in' '68.8 in' '77 in' '53.9 in' '53.2 in' '63.8 in' '49.9 in'
 '75.2 in' '76.8 in' '78.3 in' '74.3 in' '62.4 in' '--' '99.2 in'
 '47.7 in' '45.8 in' '52.6 in' '52 in' '61.9 in' '59.9 in' '64 in'
 '107.7 in' '88 in' '51.3 in' '51.6 in' '50.1 in' '83.6 in' '77.3 in'
```

```
 '48.4 in' '60.6 in' '49 in' '51.7 in' '61.4 in' '84.1 in' '53.6 in'
 '46.6 in' '48.9 in' '74.8 in' '78.7 in' '52.2 in' '82.6 in' '47.1 in'
 '47.4 in' '47 in' '50.2 in' '50.3 in' '83.7 in' '46 in' '72.1 in'
 '52.7 in' '49.2 in' '84.6 in' '84.8 in' '46.3 in' '69.2 in' '79.3 in'
 '75.4 in' '53.8 in' '47.2 in' '52.3 in' '46.8 in' '48 in' '45.2 in'
 '44.3 in' '46.7 in' '49.4 in' '77.9 in' '80.2 in' '83.9 in' '82.2 in'
 '100.7 in' '46.4 in' '73.1 in' '74.5 in' '47.5 in' '49.7 in' '62.8 in'
 '73.2 in' '64.3 in' '50.5 in' '52.4 in' '49.8 in' '44 in' '82.1 in'
 '98.7 in' '52.5 in' '49.6 in' '44.7 in' '82.9 in' '101 in' '49.5 in'
 '77.8 in' '72.9 in' '82.3 in' '77.1 in' '74.7 in' '73.3 in' '100.8 in'
 '84.3 in' '81.2 in' '82.8 in' '48.1 in' '81.6 in' '78.2 in' '85 in'
 '80.9 in' '80.7 in' '84.5 in' '62 in' '48.3 in' '46.1 in' '83.2 in'
 '82.7 in' '48.2 in' '60.3 in' '53.4 in']
['193.8 in' '184.8 in' '204.3 in' '203.7 in' '189.8 in' '175.1 in'
 '166.9 in' '181.1 in' '192.3 in' '166.6 in' '173.4 in' '201.2 in'
 '188.4 in' '200.8 in' '230 in' '183.3 in' '182.3 in' '192.5 in'
 '191.1 in' '198.3 in' '210 in' '189.9 in' '182.5 in' '192.6 in'
 '203.9 in' '176.2 in' '174.7 in' '203.6 in' '188.3 in' '167.6 in'
 '183.9 in' '184.9 in' '192.8 in' '182 in' '174.5 in' '173 in' '224.9 in'
 '198.5 in' '198.4 in' '224 in' '191.4 in' '187.4 in' '190.3 in'
 '231.7 in' '201.3 in' '186.3 in' '179.4 in' '202.8 in' '191.9 in'
 '203.8 in' '199.4 in' '179.9 in' '179.3 in' '184.5 in' '172.4 in'
 '180.9 in' '173.8 in' '184.1 in' '200 in' '192.9 in' '212.4 in'
 '194.5 in' '212.3 in' '185.1 in' '189.5 in' '190.6 in' '179.5 in'
 '224.3 in' '196.2 in' '189.6 in' '182.8 in' '176.9 in' '212.7 in'
 '185.5 in' '205 in' '190.9 in' '195.5 in' '188.8 in' '196.1 in'
 '191.3 in' '160.1 in' '183.5 in' '178.1 in' '177.3 in' '201.4 in'
 '202.9 in' '191.5 in' '181.8 in' '192.1 in' '209.3 in' '229 in'
 '162.8 in' '202.5 in' '159.6 in' '163 in' '165.2 in' '194.6 in'
 '178.7 in' '183.4 in' '197.2 in' '182.6 in' '185.7 in' '185 in'
 '182.2 in' '168 in' '182.7 in' '197.9 in' '175.4 in' '190.4 in'
 '190.7 in' '202 in' '229.9 in' '185.3 in' '189 in' '181 in' '175.8 in'
 '193.1 in' '189.1 in' '232.9 in' '195.7 in' '229.8 in' '169.1 in'
 '188.5 in' '187.2 in' '200.9 in' '186.8 in' '179.1 in' '190.2 in'
 '179.2 in' '229.5 in' '184.4 in' '161.6 in' '222.9 in' '185.6 in'
 '204 in' '183.6 in' '183.1 in' '182.1 in' '193.9 in' '171.9 in'
 '172.6 in' '180.7 in' '185.9 in' '199.6 in' '176.4 in' '231.9 in'
 '198.6 in' '174.8 in' '189.2 in' '180.2 in' '200.2 in' '205.8 in'
 '180 in' '197.1 in' '198 in' '180.3 in' '168.3 in' '181.9 in' '175.6 in'
 '181.5 in' '178.3 in' '190 in' '191.8 in' '199.8 in' '180.6 in'
 '168.4 in' '188.9 in' '202.2 in' '187.8 in' '173.2 in' '196.5 in'
 '172.8 in' '200.6 in' '191.6 in' '177.8 in' '184.6 in' '204.9 in'
 '208.3 in' '185.2 in' '173.6 in' '192.2 in' '177.9 in' '183.8 in'
 '162 in' '168.7 in' '193.5 in' '189.7 in' '194.2 in' '164 in' '203.2 in'
 '182.9 in' '202.4 in' '170.9 in' '188.1 in' '175.5 in' '187.1 in'
 '224.4 in' '174.3 in' '186.7 in' '184.2 in' '193.3 in' '159.3 in'
 '152.8 in' '167 in' '205.6 in' '195 in' '172 in' '191.7 in' '190.5 in'
 '175.9 in' '205.5 in' '204.1 in' '201.8 in' '176.5 in' '221.3 in'
```

'197.5 in' '194.9 in' '215 in' '200.1 in' '192.4 in' '237.9 in'
'199.9 in' '193.6 in' '241.8 in' '218 in' '221.9 in' '194.1 in'
'230.2 in' '166.7 in' '194.7 in' '202.1 in' '200.3 in' '191 in'
'196.8 in' '193.4 in' '183.7 in' '208.9 in' '184.3 in' '164.3 in'
'198.8 in' '181.3 in' '167.2 in' '161.8 in' '186.1 in' '171.2 in'
'195.6 in' '169.3 in' '171.3 in' '195.3 in' '170.5 in' '228.9 in'
'196.9 in' '207.4 in' '206.5 in' '179 in' '176 in' '193.2 in' '206.6 in'
'183 in' '178.8 in' '171.7 in' '197.7 in' '162.4 in' '146.2 in'
'194.3 in' '205.3 in' '161.3 in' '171.6 in' '200.4 in' '157.4 in'
'228.7 in' '172.2 in' '192.7 in' '196.6 in' '167.5 in' '160 in'
'163.7 in' '154.1 in' '164.9 in' '195.4 in' '174.2 in' '192 in'
'201.9 in' '170.4 in' '161.7 in' '168.2 in' '196.4 in' '243.7 in'
'229.4 in' '155.5 in' '149 in' '235.5 in' '201.6 in' '225.5 in'
'200.7 in' '208.1 in' '207.2 in' '193.7 in' '224.6 in' '151.9 in'
'190.1 in' '188.2 in' '186.2 in' '180.5 in' '171.5 in' '216.7 in'
'166.8 in' '177 in' '222.3 in' '169.8 in' '178.2 in' '161.4 in'
'208.5 in' '220.9 in' '190.8 in' '202.6 in' '176.8 in' '212.6 in'
'154.7 in' '208.6 in' '177.5 in' '194 in' '207.9 in' '220.8 in'
'203.3 in' '219.3 in' '177.1 in' '178.5 in' '197 in' '180.1 in'
'194.8 in' '173.9 in' '194.4 in' '184.7 in' '206.9 in' '191.2 in'
'228.1 in' '207.6 in' '168.8 in' '195.1 in' '169.6 in' '174.6 in'
'172.1 in' '144.4 in' '169 in' '198.9 in' '160.7 in' '188 in' '237.6 in'
'195.9 in' '201.7 in' '207.1 in' '209 in' '165 in' '187 in' '196.7 in'
'176.3 in' '187.5 in' '207 in' '206 in' '159 in' '207.7 in' '178.6 in'
'195.2 in' '205.1 in' '175 in' '229.3 in' '150.9 in' '179.6 in'
'162.2 in' '186 in' '210.2 in' '158 in' '176.6 in' '164.7 in' '186.5 in'
'208.7 in' '197.3 in' '182.4 in' '204.4 in' '145.6 in' '159.7 in'
'204.7 in' '206.1 in' '184 in' '222.4 in' '239.6 in' '212 in' '152.7 in'
'227.7 in' '169.5 in' '201 in' '223.8 in' '--' '176.7 in' '179.7 in'
'173.5 in' '189.3 in' '203 in' '187.6 in' '173.1 in' '181.7 in'
'170.7 in' '231.8 in' '186.4 in' '187.9 in' '151.1 in' '219.5 in'
'169.9 in' '139.6 in' '193 in' '149.4 in' '263.9 in' '187.3 in'
'196.3 in' '150.6 in' '180.4 in' '228.5 in' '185.8 in' '186.6 in'
'219.9 in' '211.2 in' '172.3 in' '201.5 in' '207.8 in' '183.2 in'
'171 in' '225.9 in' '166.1 in' '177.4 in' '227.9 in' '250.5 in'
'161.5 in' '213.2 in' '165.9 in' '197.6 in' '197.4 in' '175.2 in'
'167.8 in' '171.1 in' '181.6 in' '206.2 in' '188.7 in' '163.3 in'
'244.1 in' '179.8 in' '165.3 in' '150.1 in' '220.6 in' '227.5 in'
'242.8 in' '168.9 in' '231 in' '177.2 in' '209.1 in' '174.4 in'
'177.6 in' '148.8 in' '206.7 in' '167.3 in' '211.4 in' '203.1 in'
'156.7 in' '199.1 in' '180.8 in' '147 in' '219.4 in' '155.3 in'
'224.1 in' '235.8 in' '165.8 in' '185.4 in' '178.4 in' '214.7 in'
'178.9 in' '165.4 in' '171.4 in' '159.4 in' '144.7 in' '218.3 in'
'155 in' '157 in' '250.3 in' '157.2 in' '168.1 in' '159.8 in' '221.6 in'
'174.1 in' '211.1 in' '215.4 in' '163.1 in' '206.8 in' '174.9 in'
'169.7 in' '201.1 in' '164.5 in' '155.4 in' '166.5 in' '200.5 in'
'232.1 in' '176.1 in' '166 in' '236.7 in' '247.8 in' '168.5 in' '174 in'
'209.8 in' '170.3 in' '161.1 in' '106.1 in' '163.2 in' '169.4 in'

```
 '167.7 in' '163.8 in' '199.2 in' '189.4 in' '211.9 in' '154.3 in'
 '160.5 in' '243.9 in' '226.2 in' '164.8 in' '225.8 in' '172.5 in'
 '208.4 in' '199.3 in' '198.1 in' '167.4 in' '167.9 in' '249.2 in'
 '218.5 in' '205.9 in' '230.1 in' '153.5 in' '163.5 in' '159.1 in'
 '213.1 in' '215.1 in' '178 in' '217.5 in' '175.7 in' '196 in' '207.3 in'
 '195.8 in' '215.3 in' '198.2 in' '218.8 in' '166.2 in' '158.5 in'
 '146.8 in' '229.6 in' '239 in' '217.8 in' '154.9 in' '224.2 in'
 '205.7 in' '152.5 in' '172.9 in' '143.1 in' '155.8 in' '172.7 in'
 '247.6 in' '222 in' '175.3 in' '236 in' '158.1 in' '224.5 in' '266.1 in'
 '162.9 in' '250.4 in' '188.6 in' '143.9 in' '202.7 in' '225.1 in'
 '152 in' '223.3 in' '220.1 in' '157.3 in' '202.3 in' '247.9 in'
 '237.2 in' '157.1 in' '148 in' '177.7 in' '164.4 in' '186.9 in' '199 in'
 '170.2 in' '156.1 in' '165.1 in' '221.4 in' '157.9 in' '218.9 in'
 '155.7 in' '206.3 in' '211.8 in' '243.2 in' '199.7 in' '157.5 in'
 '220.2 in' '173.7 in' '181.2 in' '153 in' '211.5 in' '222.1 in'
 '241.2 in' '246.6 in' '210.7 in' '228.2 in' '214.8 in' '237.3 in'
 '221.7 in' '151.8 in' '236.4 in' '170.1 in' '155.9 in' '214.1 in'
 '249 in' '197.8 in' '160.4 in' '227.6 in' '161.9 in' '246.7 in'
 '229.7 in' '204.8 in' '220.3 in' '239.7 in' '209.7 in' '241.3 in'
 '215.8 in' '142.8 in' '164.6 in' '152.6 in' '205.4 in' '238.9 in'
 '208 in' '147.2 in' '205.2 in' '226.9 in' '227.2 in' '147.7 in'
 '212.2 in' '207.5 in']
['5 seats' '4 seats' '8 seats' '7 seats' '6 seats' '2 seats' '3 seats'
 '15 seats' '12 seats' '9 seats' '--' '10 seats']
['73 in' '81.5 in' '78.6 in' '78.5 in' '84.8 in' '71.4 in' '70.9 in'
 '83.5 in' '73.6 in' '79.6 in' '82.8 in' '85.5 in' '73.8 in' '75 in'
 '80 in' '70.8 in' '74.6 in' '82.5 in' '73.2 in' '90.2 in' '93.8 in'
 '81.3 in' '71.8 in' '84.3 in' '80.5 in' '72.8 in' '78.1 in' '90.4 in'
 '72.4 in' '74.7 in' '69.9 in' '73.7 in' '71.2 in' '83.9 in' '77.2 in'
 '76.7 in' '74.4 in' '75.2 in' '81.9 in' '81.2 in' '85.6 in' '69 in'
 '88.5 in' '72 in' '77.5 in' '79.5 in' '72.3 in' '78.9 in' '73.4 in'
 '79.3 in' '69.2 in' '72.9 in' '72.2 in' '80.3 in' '75.5 in' '70.1 in'
 '77.7 in' '71.1 in' '71.7 in' '80.8 in' '85.8 in' '72.6 in' '72.5 in'
 '67.8 in' '81.8 in' '89.3 in' '91.5 in' '82.6 in' '96.8 in' '79.4 in'
 '79 in' '68.5 in' '83.7 in' '87.5 in' '74 in' '70 in' '70.5 in' '74.9 in'
 '81.1 in' '79.9 in' '70.7 in' '74.1 in' '85.7 in' '82.1 in' '69.8 in'
 '80.1 in' '81 in' '74.8 in' '73.1 in' '84.1 in' '78.4 in' '77.3 in'
 '71 in' '66.7 in' '79.1 in' '82.2 in' '83.3 in' '69.5 in' '75.8 in'
 '67.9 in' '84.2 in' '87.1 in' '77.8 in' '69.6 in' '70.6 in' '69.1 in'
 '74.2 in' '71.6 in' '76.2 in' '69.3 in' '83.1 in' '71.3 in' '71.5 in'
 '75.4 in' '76 in' '80.9 in' '66.9 in' '92.3 in' '78 in' '97 in' '86.1 in'
 '79.2 in' '82.3 in' '73.9 in' '88.8 in' '83.4 in' '85.4 in' '75.9 in'
 '93.4 in' '73.3 in' '80.6 in' '77.1 in' '87.4 in' '79.8 in' '78.8 in'
 '78.3 in' '74.5 in' '84.9 in' '91.8 in' '70.4 in' '86 in' '86.5 in'
 '81.4 in' '75.7 in' '87.3 in' '84 in' '66.2 in' '82 in' '86.2 in' '67 in'
 '68.3 in' '69.4 in' '85.2 in' '73.5 in' '77.6 in' '86.7 in' '79.7 in'
 '69.7 in' '68.9 in' '72.7 in' '87.7 in' '97.4 in' '82.7 in' '78.2 in'
 '76.1 in' '80.2 in' '68.1 in' '95.7 in' '82.9 in' '75.6 in' '83 in'
```

```
 '82.4 in' '86.4 in' '75.1 in' '88 in' '76.3 in' '77.9 in' '85.9 in'
 '86.9 in' '83.2 in' '85 in' '71.9 in' '84.4 in' '68.2 in' '88.3 in'
 '80.4 in' '68.6 in' '70.3 in' '72.1 in' '78.7 in' '81.7 in' '77.4 in'
 '76.8 in' '68.7 in' '67.6 in' '84.6 in' '74.3 in' '70.2 in' '65.7 in'
 '--' '67.5 in' '68 in' '85.1 in' '65.6 in' '80.7 in' '98.6 in' '76.6 in'
 '68.8 in' '76.5 in' '75.3 in' '76.4 in' '77 in' '65.4 in' '67.1 in'
 '66 in' '62.9 in' '67.3 in' '83.8 in' '67.2 in' '86.3 in' '76.9 in'
 '61.4 in' '87.6 in' '67.4 in' '66.1 in' '66.5 in' '89.9 in' '66.3 in'
 '84.5 in' '64.1 in' '87.2 in' '65.9 in' '66.4 in' '65 in' '89.1 in'
 '98 in' '67.7 in' '66.6 in' '89.2 in' '89.5 in' '64 in' '68.4 in'
 '66.8 in' '65.3 in' '63.7 in' '81.6 in' '62.6 in']
```

Regarding fuel_tank_volume and maximum_seating, we can see that there appears to be a pattern in the suffixes, "gal" and "seats" accordingly. We will now remove them and then convert all values to numeric.

removing unecesary string values in columns then cleaning up any values that contain '–' and replacing it with NaN Lastly, converting the value first to a string type and then to a float type

[27]:
```
df_cln_2['fuel_tank_volume']=df_cln_2['fuel_tank_volume'].astype(str).str.
 ↪replace(' gal', '').replace('--',np.NaN).astype(float)
df_cln_2['height']=df_cln_2['height'].astype(str).str.replace(' in', '').
 ↪replace('--',np.NaN).astype(float)
df_cln_2['length']=df_cln_2['height'].astype(str).str.replace(' in', '').
 ↪replace('--',np.NaN).astype(float)
df_cln_2['maximum_seating']=df_cln_2['maximum_seating'].astype(str).str.
 ↪replace(' seats', '').replace('--',np.NaN).astype(float)
df_cln_2['width']=df_cln_2['width'].astype(str).str.replace(' in', '').
 ↪replace('--',np.NaN).astype(float)
```

doing a quick profile on the subsetted columns

[28]:
```
print(df_cln_2.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 697989 entries, 38 to 3000039
Data columns (total 35 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   vin                   697989 non-null  object
 1   body_type             697989 non-null  object
 2   city                  697989 non-null  object
 3   city_fuel_economy     697989 non-null  float64
 4   daysonmarket          697989 non-null  int64
 5   engine_displacement   697989 non-null  float64
 6   engine_type           697989 non-null  object
 7   exterior_color        697989 non-null  object
 8   frame_damaged         697989 non-null  object
 9   franchise_dealer      697989 non-null  bool
```

```
 10  franchise_make         697989 non-null  object
 11  fuel_tank_volume       697819 non-null  float64
 12  fuel_type              697989 non-null  object
 13  has_accidents          697989 non-null  object
 14  height                 697882 non-null  float64
 15  highway_fuel_economy   697989 non-null  float64
 16  horsepower             697989 non-null  float64
 17  isCab                  697989 non-null  object
 18  is_new                 697989 non-null  bool
 19  length                 697882 non-null  float64
 20  listing_color          697989 non-null  object
 21  make_name              697989 non-null  object
 22  maximum_seating        697884 non-null  float64
 23  mileage                697989 non-null  float64
 24  model_name             697989 non-null  object
 25  owner_count            697989 non-null  float64
 26  power                  697989 non-null  object
 27  price                  697989 non-null  float64
 28  salvage                697989 non-null  object
 29  seller_rating          697989 non-null  float64
 30  torque                 697989 non-null  object
 31  transmission           697989 non-null  object
 32  wheel_system           697989 non-null  object
 33  width                  697885 non-null  float64
 34  year                   697989 non-null  int64
dtypes: bool(2), float64(13), int64(2), object(18)
memory usage: 182.4+ MB
None
```

We can now see that all of our column values have been adjusted to the correct datatypes. We will next proceed with cleaning up the remainder of our data attributes. A quick count on our datatypes shows us that we still have 19 categorical values, 14 numerical types and 2 booleans (true/false) left to work wikth. Since our ultimate goal is doing a regression and one classification model, we will next work on trimming down our 'object' categorical data types.

count of datatypes in current dataframe

[29]:
```
print(df_cln_2.dtypes.value_counts())
print("------------------------------------------")
#showing only the object type columns
print(df_cln_2.select_dtypes(include='object').columns)
```

```
object     18
float64    13
bool        2
int64       2
dtype: int64
------------------------------------------
Index(['vin', 'body_type', 'city', 'engine_type', 'exterior_color',
```

```
       'frame_damaged', 'franchise_make', 'fuel_type', 'has_accidents',
       'isCab', 'listing_color', 'make_name', 'model_name', 'power', 'salvage',
       'torque', 'transmission', 'wheel_system'],
      dtype='object')
```

After the team decided to choose 'body_type' as main the main classification term. We decided to create a final dataframe with only the attributes we found useful for our model which excludes: - **'vin'**: acts as primary index but not useful for modeling - **'city'**: no model of ours will require geographic features, also since there is not state or zip attributes, it is ambiguous - **'engine_type'**: use for EDA but 'horsepower' is the continous version of this attribute - **'franchise_dealer'**: not useful for our model - **'franchise_make'**: same as 'make_name' which has too many levels as a categorical value and not necessary for our predictin model - **'fuel_tank_volume'**: - **'fuel_type'**: - **'isCab'**: - **'listing_color'**: - **'make_name'**: - **'model_name'**: - **'power'**: 'horsepower' attribute will be used instead - **'salvage'**: the value that is associated with depreciation number which is more of accounting. In real world, we pay what is the market value of the car. - **'torque'**: 'horsepower' attribute will be used instead - **'transmission'**: - **'wheel_system'**:

[30]:
```
df_cln_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 697989 entries, 38 to 3000039
Data columns (total 35 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   vin                   697989 non-null  object
 1   body_type             697989 non-null  object
 2   city                  697989 non-null  object
 3   city_fuel_economy     697989 non-null  float64
 4   daysonmarket          697989 non-null  int64
 5   engine_displacement   697989 non-null  float64
 6   engine_type           697989 non-null  object
 7   exterior_color        697989 non-null  object
 8   frame_damaged         697989 non-null  object
 9   franchise_dealer      697989 non-null  bool
 10  franchise_make        697989 non-null  object
 11  fuel_tank_volume      697819 non-null  float64
 12  fuel_type             697989 non-null  object
 13  has_accidents         697989 non-null  object
 14  height                697882 non-null  float64
 15  highway_fuel_economy  697989 non-null  float64
 16  horsepower            697989 non-null  float64
 17  isCab                 697989 non-null  object
 18  is_new                697989 non-null  bool
 19  length                697882 non-null  float64
 20  listing_color         697989 non-null  object
 21  make_name             697989 non-null  object
 22  maximum_seating       697884 non-null  float64
 23  mileage               697989 non-null  float64
```

```
 24  model_name            697989 non-null  object
 25  owner_count           697989 non-null  float64
 26  power                 697989 non-null  object
 27  price                 697989 non-null  float64
 28  salvage               697989 non-null  object
 29  seller_rating         697989 non-null  float64
 30  torque                697989 non-null  object
 31  transmission          697989 non-null  object
 32  wheel_system          697989 non-null  object
 33  width                 697885 non-null  float64
 34  year                  697989 non-null  int64
dtypes: bool(2), float64(13), int64(2), object(18)
memory usage: 182.4+ MB
```

creating final df for analysis

[31]:
```
df_final = df_cln_2.iloc[:, np.r_[1,3:6,8,13,14:17,18:20,22:24,25,27,29,33:35]]
print(df_final.info())
print(df_final.dtypes.value_counts())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 697989 entries, 38 to 3000039
Data columns (total 18 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   body_type             697989 non-null  object
 1   city_fuel_economy     697989 non-null  float64
 2   daysonmarket          697989 non-null  int64
 3   engine_displacement   697989 non-null  float64
 4   frame_damaged         697989 non-null  object
 5   has_accidents         697989 non-null  object
 6   height                697882 non-null  float64
 7   highway_fuel_economy  697989 non-null  float64
 8   horsepower            697989 non-null  float64
 9   is_new                697989 non-null  bool
 10  length                697882 non-null  float64
 11  maximum_seating       697884 non-null  float64
 12  mileage               697989 non-null  float64
 13  owner_count           697989 non-null  float64
 14  price                 697989 non-null  float64
 15  seller_rating         697989 non-null  float64
 16  width                 697885 non-null  float64
 17  year                  697989 non-null  int64
dtypes: bool(1), float64(12), int64(2), object(3)
memory usage: 96.5+ MB
None
float64    12
object      3
```

```
int64       2
bool        1
dtype: int64
```

1.0.6 Simple Statistics

Look at the final data, we have total 18 columns, their data types are: bool(1), float64(12), int64(2), object(3). As our main goal is to predict car price and car type,in the simple statistics we need to find out the 5 most significant attributes that affect the car price, which's divided to below 5 steps: 1.Check the data range, mode, mean, median, variance and counts, etc 2.Check data normality 3.Correlation check 4.With the result from correlation checking,the 5 most significan attributes are horsepower(0.629),mileage(-0.43),highway_fuel_economy(-0.401), year(0.371) and engine_displacement(0.427) An intresting thing we found is that the seller_rating and owner_count don't show strong relationship to car price from correlation checking, they're only -0.219 and -0.035

	engine_displacement	highway_fuel_economy	horsepower	is_new	maximum_seating	mileage	owner_count	price	seller_rating	year
city_fuel_economy	-0.783	0.943	-0.778	-0.011	-0.399	-0.172	-0.12	-0.369	-0.047	0.173
engine_displacement		-0.737	0.832	-0.014	0.378	0.116	0.078	0.427	0.007	-0.107
highway_fuel_economy			-0.727	-0.015	-0.409	-0.139	-0.078	-0.401	-0.044	0.129
horsepower				0.002	0.306	-0.029	0.004	0.629	0.046	0.041
is_new					0.016	-0.026	-0.013	0.031		0.028
maximum_seating						0.075	-0.054	0.131	0.009	0.086
mileage							0.476	-0.43	0.013	-0.719
owner_count								-0.219	-0.002	-0.529
price									0.086	0.371
seller_rating										-0.035

Median, Standard Deviation, Mean in a Pandas Dataframe We can see from observing the simple statistic is that the median price aligns with the IQR as most of the cars are priced at around $21,700. However, what's interesting to note here is that the though the median and the mean are somewhat similar, there is a standard deviation of about \$15,878. This could be due to the fact that the extremities in car pricing varies greatly from the average with the lost car priced at \$484 while the highest price car is at \$3,299,995

[32]:
```
print(df_final['price'].aggregate([np.median, np.std, np.mean]).reset_index())
print('min: '+str(df_final['price'].min()))
print('min: '+str(df_final['price'].max()))
```

```
    index         price
0  median  21700.000000
1     std  15878.458032
2    mean  24434.421728
min: 484.0
min: 3299995.0
```

Inter-quartile Range

[33]:
```
print(df_final.describe()['price'][['25%', '50%', '75%']])
print(df_final.max()['price'])
```

```
25%    16230.0
50%    21700.0
75%    29999.0
Name: price, dtype: float64
3299995.0
```

We can also see that our dataset has the majority (~99%) of our cars being used cars as described in the title, with less than 1% being new.

Check the count of new cars

[34]:
```
print('count of new cars: '+str(df_final['is_new'].values.sum()))
print('count of used cars: '+str((~df_final['is_new']).values.sum()))
```

```
count of new cars: 667
count of used cars: 697322
```

We then looked into the number of different types'cars that have accidents or no accidents. We found that 84706 cars have accidents and out of that, Suv/Crossover and Sedan cars have higher numbers of accidents, as their count are larger too. Perhaps we could say love SUV and Sedan cars more than others.

check the count of cars have accidents

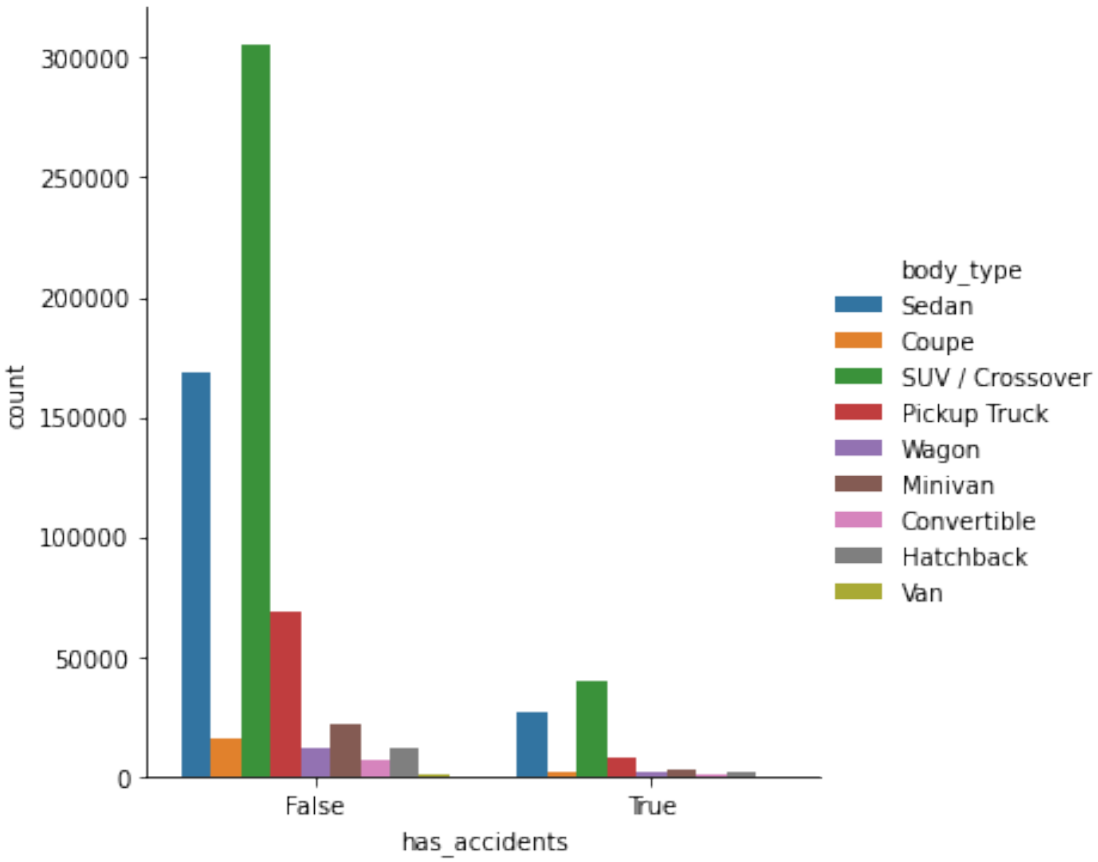
[35]:
```
print(df_final['has_accidents'].values.sum())
```

```
84706
```

Visulize the number of different type of cars that have accidents or not.

[36]:
```
sns.catplot(x="has_accidents", kind="count",hue='body_type', data=df_final)
```

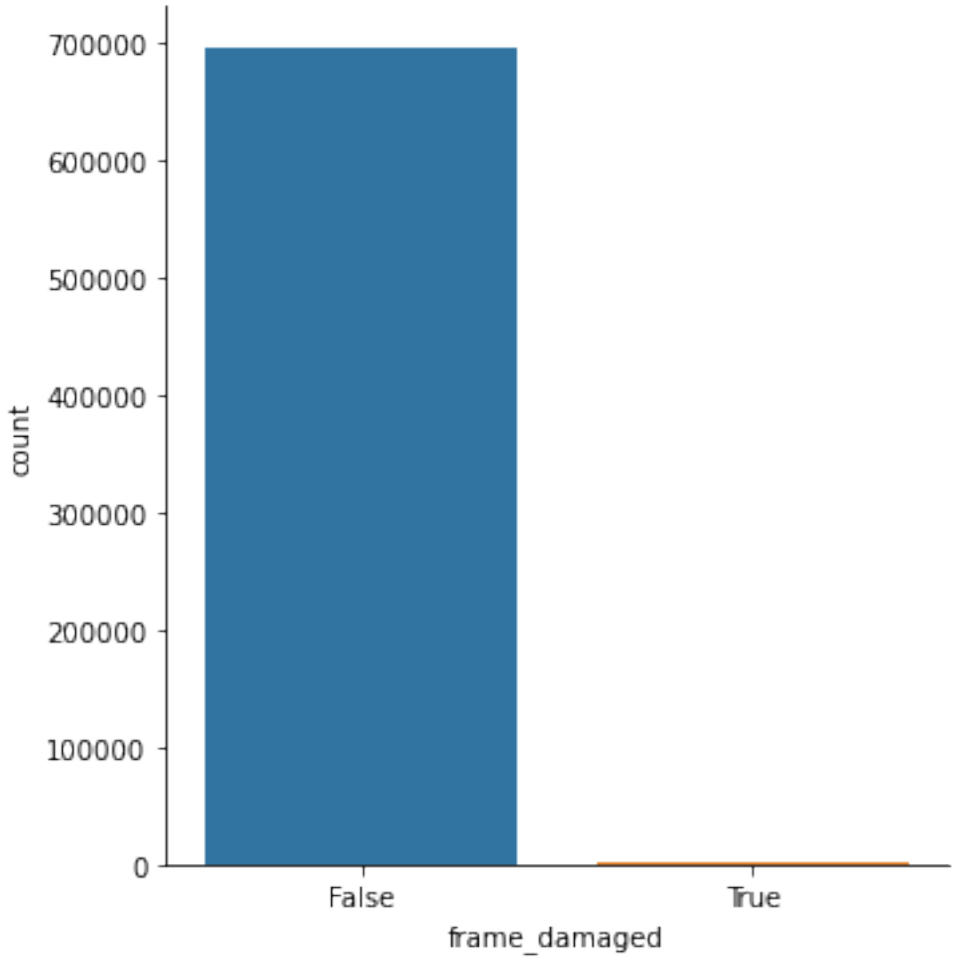
[36]:
```
<seaborn.axisgrid.FacetGrid at 0x7fa1e6c3a790>
```



Visualize the number of frame damaged cars

[37]:
```
sns.catplot(x="frame_damaged", kind="count", data=df_final)
```

[37]:
```
<seaborn.axisgrid.FacetGrid at 0x7fa476c32880>
```



check the count of frame_damaged cars

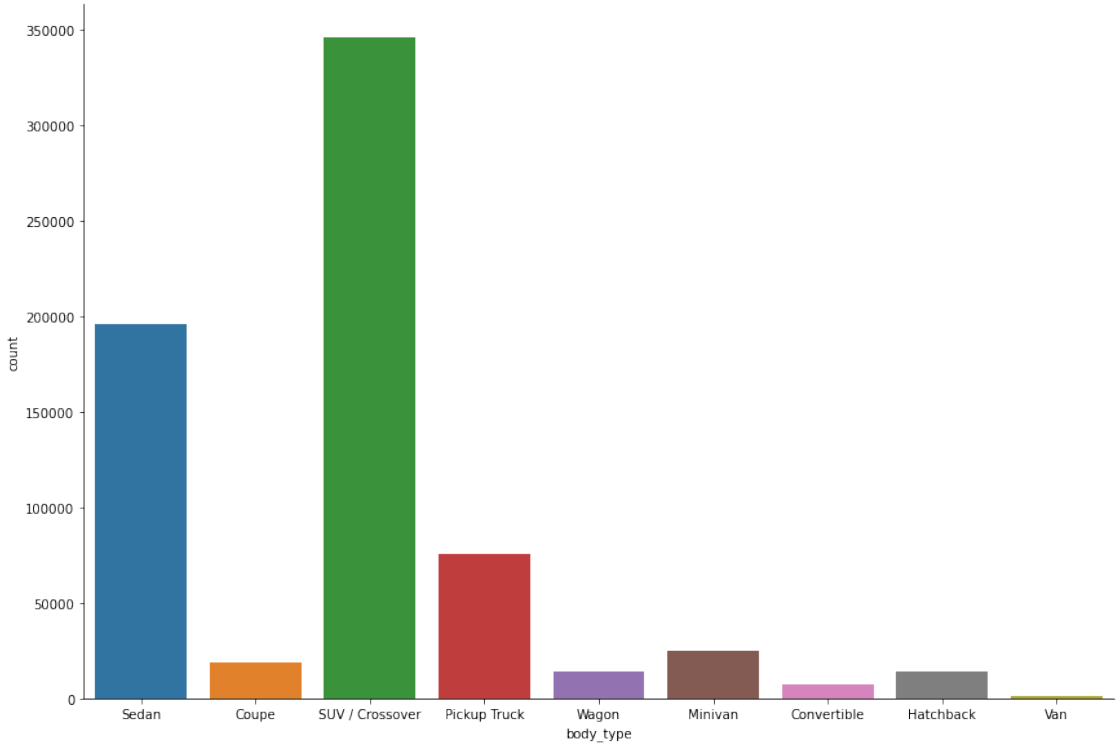
[38]:
```
df_final['frame_damaged'].values.sum()
```

[38]:
```
1587
```

check the number of different type of cars,it looks like SUV is the most popular car with almost 3500, Sedan is the 2nd most popular car.

[39]:
```
sns.catplot(x="body_type", kind="count",data=df_final,height=8, aspect=1.5)
```

[39]:
```
<seaborn.axisgrid.FacetGrid at 0x7fa0536209a0>
```



1.0.7 Data normality check

select all numberical data type from the final data

[40]:
```
df_final_num=df_final.iloc[:, np.r_[1:4,6:9,10:18]]
print(df_final_num.info())
```

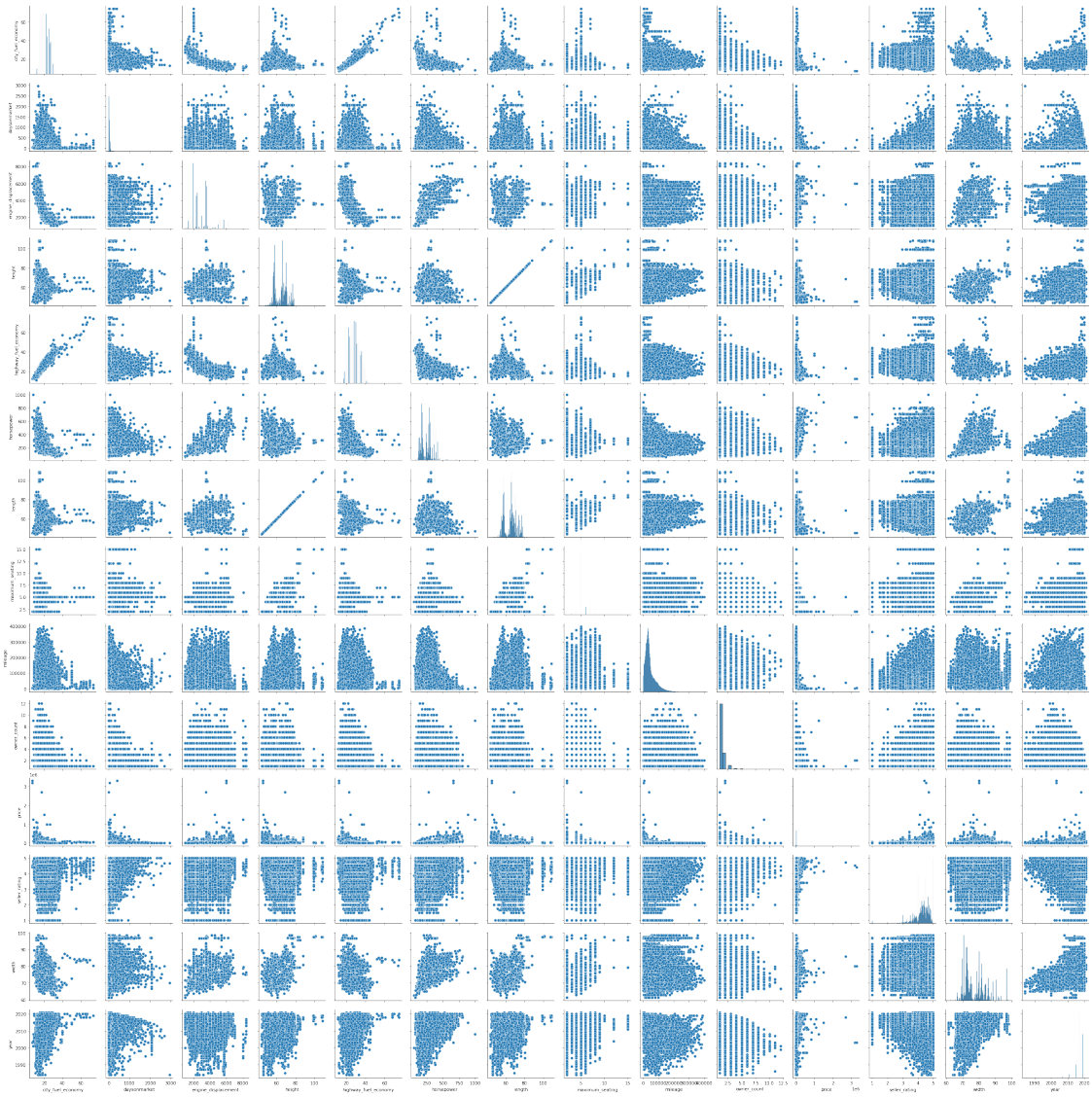
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 697989 entries, 38 to 3000039
Data columns (total 14 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   city_fuel_economy     697989 non-null  float64
 1   daysonmarket          697989 non-null  int64
 2   engine_displacement   697989 non-null  float64
 3   height                697882 non-null  float64
 4   highway_fuel_economy  697989 non-null  float64
 5   horsepower            697989 non-null  float64
 6   length                697882 non-null  float64
 7   maximum_seating       697884 non-null  float64
 8   mileage               697989 non-null  float64
 9   owner_count           697989 non-null  float64
 10  price                 697989 non-null  float64
 11  seller_rating         697989 non-null  float64
```

```
 12  width                  697885 non-null  float64
 13  year                   697989 non-null  int64
dtypes: float64(12), int64(2)
memory usage: 99.9 MB
None
```

check all numerical type data's normality with pairplot

[41]:
```
sns.pairplot(df_final_num)
```

[41]:
```
<seaborn.axisgrid.PairGrid at 0x7fa04f88da30>
```



From the histogram, we may easily see city_fuel_economy,engine_displacement and highway_fuel_economy more normally distributed than other columns. As our sam-

ple size is large enough,we use Central Limit Theorem rule and assume they're approximately normal.

1.0.8 Correlation check

data correlation with Spearman method

[42]:
```
df_final.corr(method='spearman')
```

[42]:

	city_fuel_economy	daysonmarket	engine_displacement	height	highway_fuel_economy	horsepower	is_new	length	maximum_seating	mileage	owner_count	price	seller_rating	width	year
city_fuel_economy	1.000000	0.014875	-0.853374	-0.659119	0.944451	-0.841248	0.004177	-0.659119	-0.479300	-0.142631	-0.110240	-0.447296	-0.055476	-0.627285	0.120279
daysonmarket	0.014875	1.000000	-0.036679	-0.002360	0.012493	-0.003996	0.039442	-0.002360	0.015586	-0.112199	-0.073059	0.024340	0.023484	0.046448	0.133165
engine_displacement	-0.853374	-0.036679	1.000000	0.584641	-0.775994	0.805837	-0.010572	0.584641	0.454605	0.119451	0.083177	0.424779	0.021255	0.487981	-0.072977
height	-0.659119	-0.002360	0.584641	1.000000	-0.756926	0.508703	0.000892	1.000000	0.648339	0.068104	-0.045746	0.394267	0.022718	0.527495	0.053203
highway_fuel_economy	0.944451	0.012493	-0.775994	-0.756926	1.000000	-0.770057	0.000051	-0.756926	-0.489743	-0.094535	-0.062339	-0.484108	-0.053771	-0.588123	0.058447
horsepower	-0.841248	-0.003996	0.805837	0.508703	-0.770057	1.000000	-0.001190	0.508703	0.394334	-0.021400	0.004498	0.646095	0.062560	0.662796	0.044572
is_new	0.004177	0.039442	-0.010572	0.000892	0.000051	-0.001190	1.000000	0.000892	0.001580	-0.048855	-0.013954	0.030439	0.002316	0.007194	0.040840
length	-0.659119	-0.002360	0.584641	1.000000	-0.756926	0.508703	0.000892	1.000000	0.648339	0.068104	-0.045746	0.394267	0.022718	0.527495	0.053203
maximum_seating	-0.479300	0.015586	0.454605	0.648339	-0.489743	0.394334	0.001580	0.648339	1.000000	0.068104	-0.045746	0.394267	0.022718	0.527495	0.053203
mileage	-0.142631	-0.112199	0.119451	0.068104	-0.094535	-0.021400	-0.048855	0.068104	0.068104	1.000000	-0.045746	0.394267	0.022718	0.527495	0.053203
owner_count	-0.110240	-0.073059	0.083177	-0.045746	-0.062339	0.004498	-0.013954	-0.045746	-0.045746	-0.045746	1.000000	0.394267	0.022718	0.527495	0.053203
price	-0.447296	0.024340	0.424779	0.394267	-0.484108	0.646095	0.030439	0.394267	0.394267	0.394267	0.394267	1.000000	0.022718	0.527495	0.053203
seller_rating	-0.055476	0.023484	0.021255	0.022718	-0.053771	0.062560	0.002316	0.022718	0.022718	0.022718	0.022718	0.022718	1.000000	0.527495	0.053203
width	-0.627285	0.046448	0.487981	0.527495	-0.588123	0.662796	0.007194	0.527495	0.527495	0.527495	0.527495	0.527495	0.527495	1.000000	0.053203
year	0.120279	0.133165	-0.072977	0.053203	0.058447	0.044572	0.040840	0.053203	0.053203	0.053203	0.053203	0.053203	0.053203	0.053203	1.000000

```
highway_fuel_economy -0.756926        -0.489743 -0.094535    -0.062339
horsepower            0.508703         0.394334 -0.021400     0.004498
is_new                0.000892         0.001580 -0.048855    -0.013954
length                1.000000         0.648339  0.068104    -0.045746
maximum_seating       0.648339         1.000000  0.094093    -0.038994
mileage               0.068104         0.094093  1.000000     0.467813
owner_count          -0.045746        -0.038994  0.467813     1.000000
price                 0.394267         0.236177 -0.524626    -0.331509
seller_rating         0.022718         0.010823 -0.006875    -0.001585
width                 0.527495         0.373197 -0.039158    -0.041600
year                  0.053203         0.053886 -0.703310    -0.490472

                          price  seller_rating     width      year
city_fuel_economy    -0.447296      -0.055476 -0.627285  0.120279
daysonmarket          0.024340       0.023484  0.046448  0.133165
engine_displacement   0.424779       0.021255  0.487981 -0.072977
height                0.394267       0.022718  0.527495  0.053203
highway_fuel_economy -0.484108      -0.053771 -0.588123  0.058447
horsepower            0.646095       0.062560  0.662796  0.044572
is_new                0.030439       0.002316  0.007194  0.040840
length                0.394267       0.022718  0.527495  0.053203
maximum_seating       0.236177       0.010823  0.373197  0.053886
mileage              -0.524626      -0.006875 -0.039158 -0.703310
owner_count          -0.331509      -0.001585 -0.041600 -0.490472
price                 1.000000       0.077142  0.492719  0.513243
seller_rating         0.077142       1.000000  0.072781 -0.022211
width                 0.492719       0.072781  1.000000  0.098481
year                  0.513243      -0.022211  0.098481  1.000000
```

data correlation check

[43]:
```
cormat = df_final.corr()
```

[44]:
```
def triang(cormat, triang='lower'):

    if triang == 'upper':
        rstri = pd.DataFrame(np.triu(cormat.values),
                             index=cormat.index,
                             columns=cormat.columns).round(3)
        rstri = rstri.iloc[:,1:]
        rstri.drop(rstri.tail(1).index, inplace=True)

    if triang == 'lower':
        rstri = pd.DataFrame(np.tril(cormat.values),
                             index=cormat.index,
                             columns=cormat.columns).round(3)
        rstri = rstri.iloc[:,:-1]
```

```
        rstri.drop(rstri.head(1).index, inplace=True)

    rstri.replace(to_replace=[0,1], value='', inplace=True)

    return(rstri)

triang(cormat, triang='upper')
```

[44]:

	daysonmarket	engine_displacement	height	\
city_fuel_economy	-0.016	-0.791	-0.617	
daysonmarket		-0.014	-0.001	
engine_displacement			0.552	
height				
highway_fuel_economy				
horsepower				
is_new				
length				
maximum_seating				
mileage				
owner_count				
price				
seller_rating				
width				

	highway_fuel_economy	horsepower	is_new	length	\
city_fuel_economy	0.942	-0.795	0.004	-0.617	
daysonmarket	-0.011	0.008	0.091	-0.001	
engine_displacement	-0.741	0.828	-0.009	0.552	
height	-0.736	0.461	0.002		
highway_fuel_economy		-0.737		-0.736	
horsepower			-0.001	0.461	
is_new				0.002	
length					
maximum_seating					
mileage					
owner_count					
price					
seller_rating					
width					

	maximum_seating	mileage	owner_count	price	seller_rating	\
city_fuel_economy	-0.414	-0.169	-0.122	-0.334	-0.062	
daysonmarket	0.012	-0.052	-0.004	0.034	0.027	
engine_displacement	0.377	0.114	0.087	0.366	0.024	
height	0.567	0.062	-0.062	0.231	0.028	
highway_fuel_economy	-0.42	-0.131	-0.08	-0.36	-0.059	
horsepower	0.301	-0.032	0.012	0.561	0.064	

```
is_new                        0.001  -0.035      -0.013  0.028         0.003
length                        0.567   0.062      -0.062  0.231         0.028
maximum_seating                       0.086      -0.046  0.092         0.013
mileage                                            0.49 -0.376         0.012
owner_count                                              -0.19         0.001
price                                                                  0.069
seller_rating
width
```

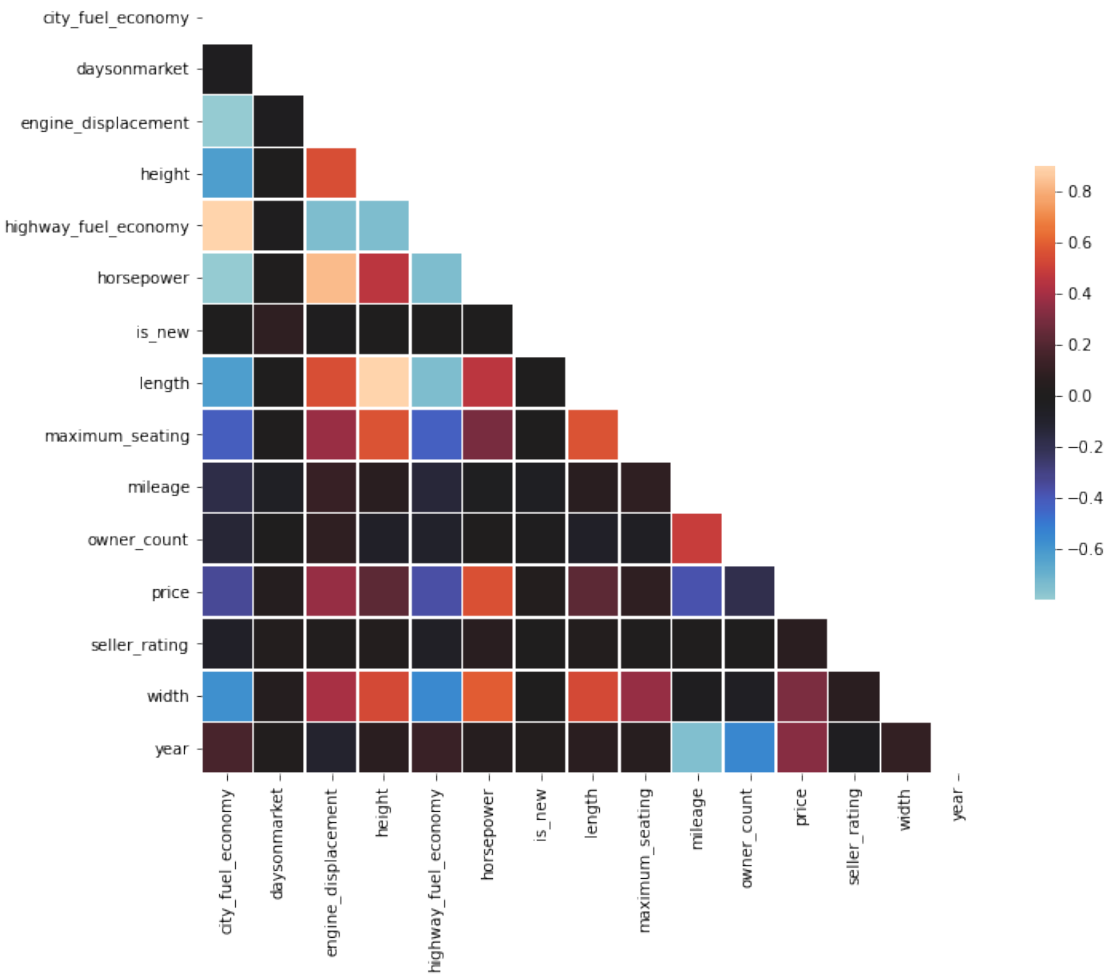
```
                       width   year
city_fuel_economy     -0.574  0.171
daysonmarket           0.039  0.023
engine_displacement    0.404 -0.097
height                  0.53  0.070
highway_fuel_economy  -0.546  0.124
horsepower             0.595  0.045
is_new                 0.006  0.029
length                  0.53  0.070
maximum_seating        0.365  0.059
mileage               -0.018 -0.746
owner_count           -0.044 -0.542
price                   0.31  0.335
seller_rating          0.069 -0.027
width                         0.103
```

Result: We may see most significan attributes are Horsepower ,Mileage,highway_fuel_economy,Year,engine_displacement

Creating the Heatmap from the Post

[45]:
```
mask = np.zeros_like(cormat, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True


fig = plt.figure(figsize=(11, 10))
sns.heatmap(cormat, mask=mask, vmax=.9, center=0,
            square=True, linewidths=.6, cbar_kws={"shrink": .5})
```

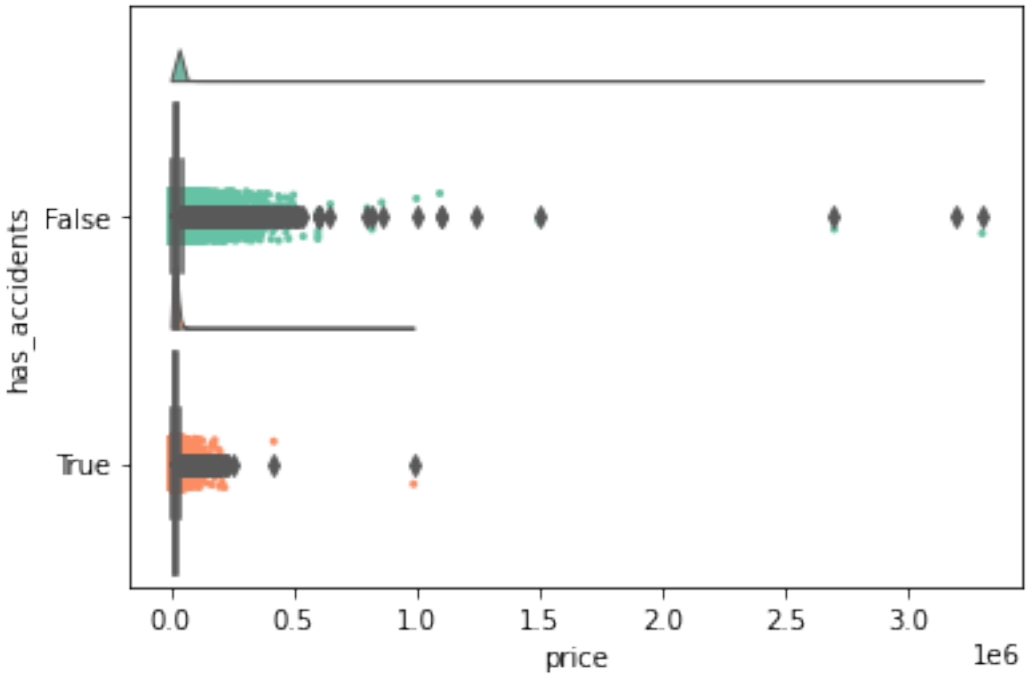
[45]:
```
<AxesSubplot:>
```



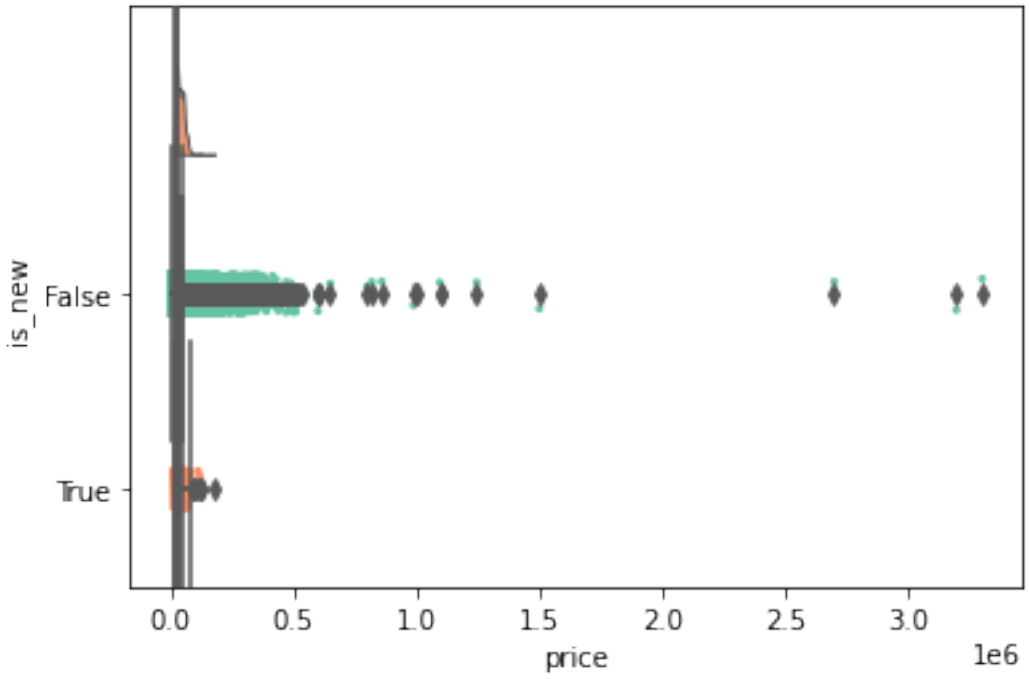
1.0.9 Visualize Attributes

1.0.10 Visualization with Raincloud Plot

[46]:
```
%matplotlib inline
ax = pt.RainCloud(x = 'has_accidents', y = 'price',
                  data = df_final,
                  width_viol =  .5,
                  width_box =  .9,
                  orient = 'h',
                  move = .0)
```



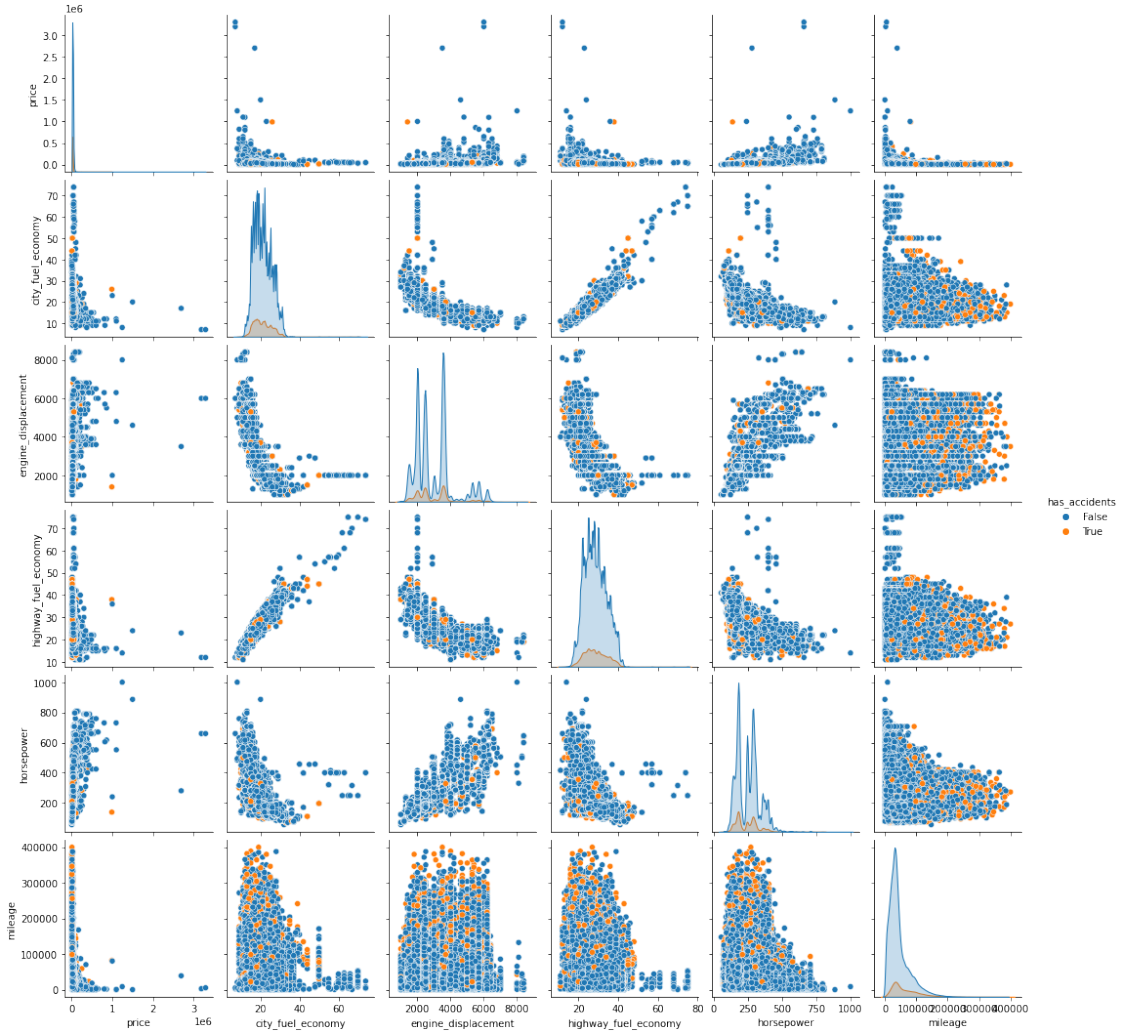
[47]:
```
ax = pt.RainCloud(x = 'is_new', y = 'price',
                  data = df_final,
                  width_viol = .9,
                  width_box = 3,
                  orient = 'h',
                  move = .0)
```



This shows the new car is more expensive than used car, the car had accident is less expensive than the car without accident

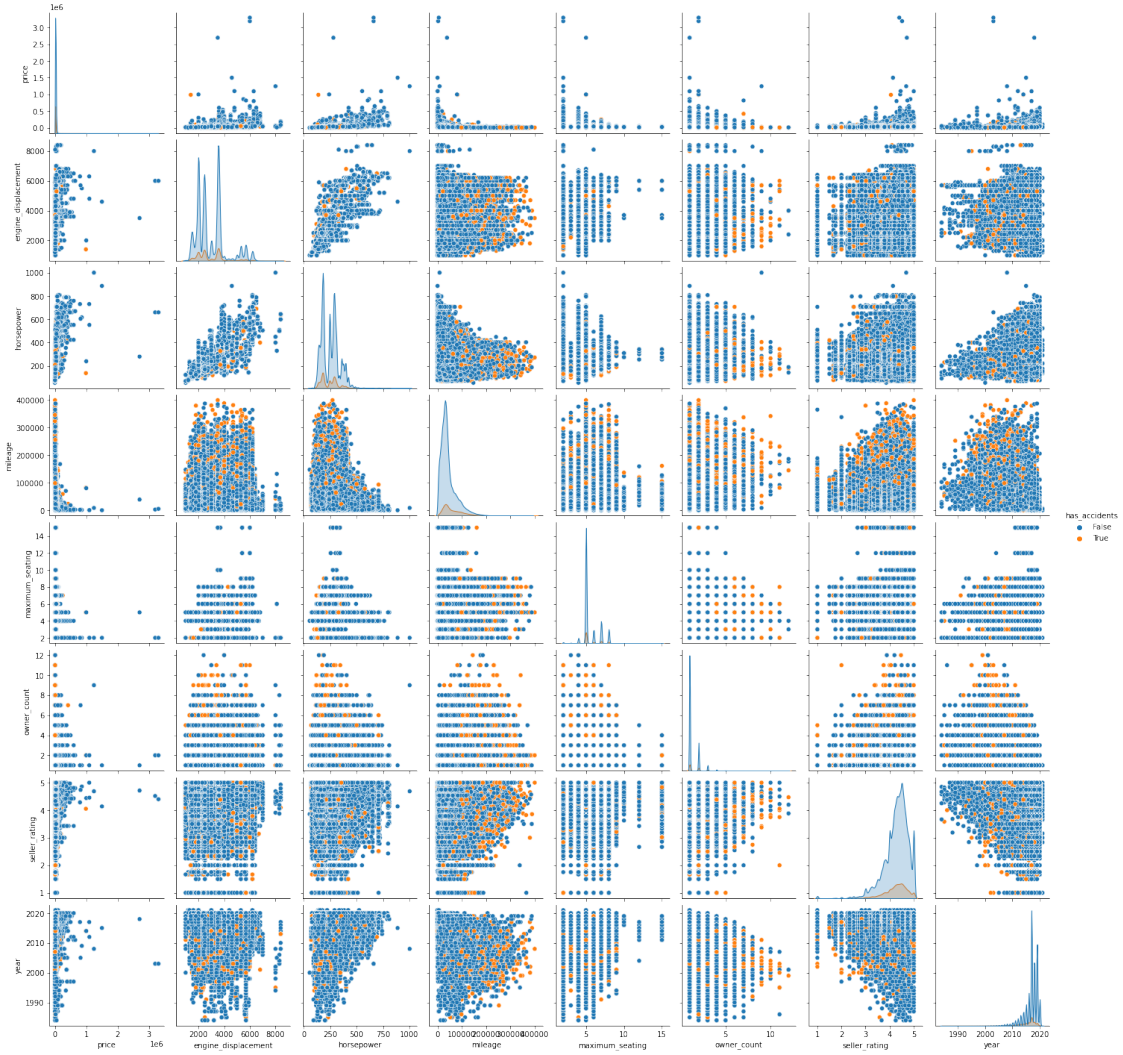
1.0.11 Visualization with Pairplot using Seaborn

[48]:
```
cols = ['price',␣
 ↪'city_fuel_economy','engine_displacement','highway_fuel_economy','horsepower','mileage']
ax = sns.pairplot(df_final, vars=cols, hue='has_accidents')
```



From above result, we rerun it again without city fuel economy and highway_fuel economy

[49]:
```
cols = ['price',␣
 ↪'engine_displacement','horsepower','mileage','maximum_seating','owner_count','seller_rating
ax = sns.pairplot(df_final, vars=cols, hue='has_accidents')
```

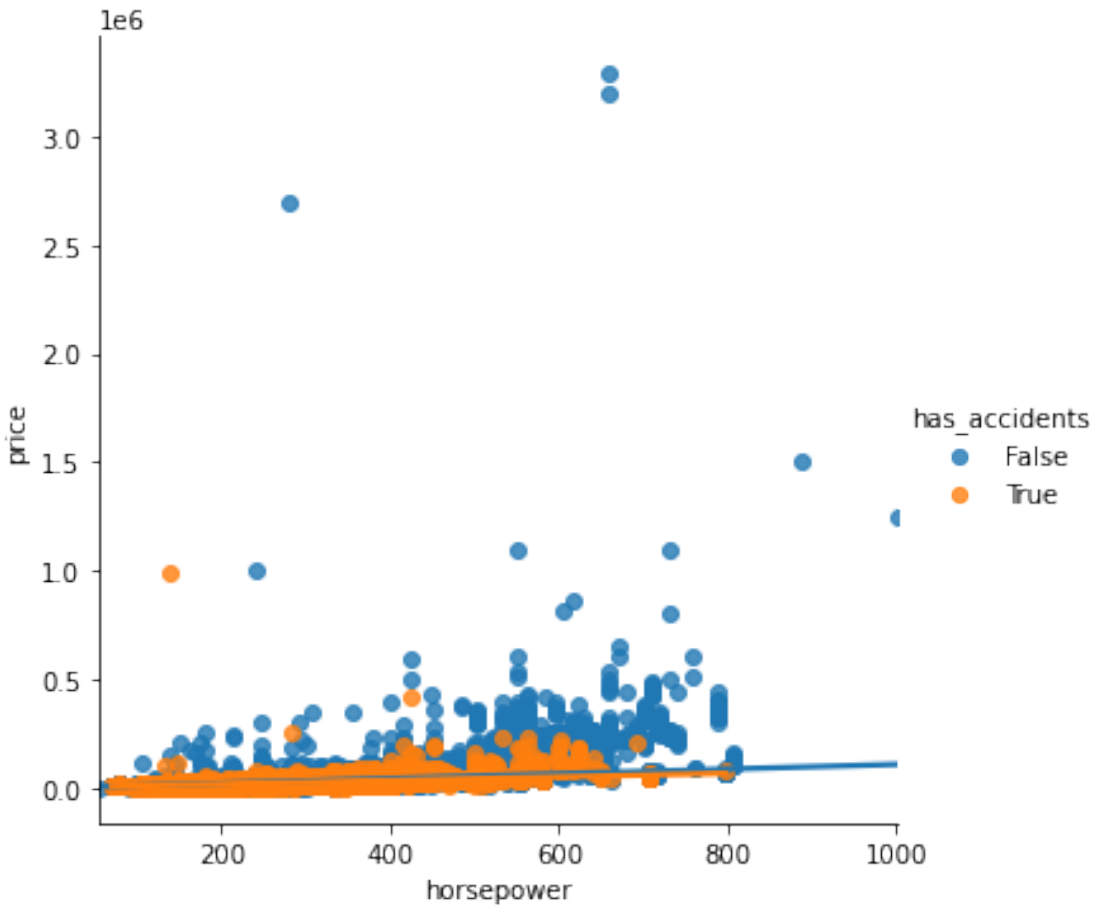


Summary: The most significant attributes to the price are horsepower, mileage, year, highway_fuel_economy, engine_displacement, possible attributes are owner_count and seller_rating.

Visualize the horsepower and price with Scatter plot using Seaborn lmplot.

[57]:
```
sns.lmplot(x='horsepower', y='price', hue='has_accidents', data=df_final)
```

[57]:
```
<seaborn.axisgrid.FacetGrid at 0x7fa12d6ff7f0>
```

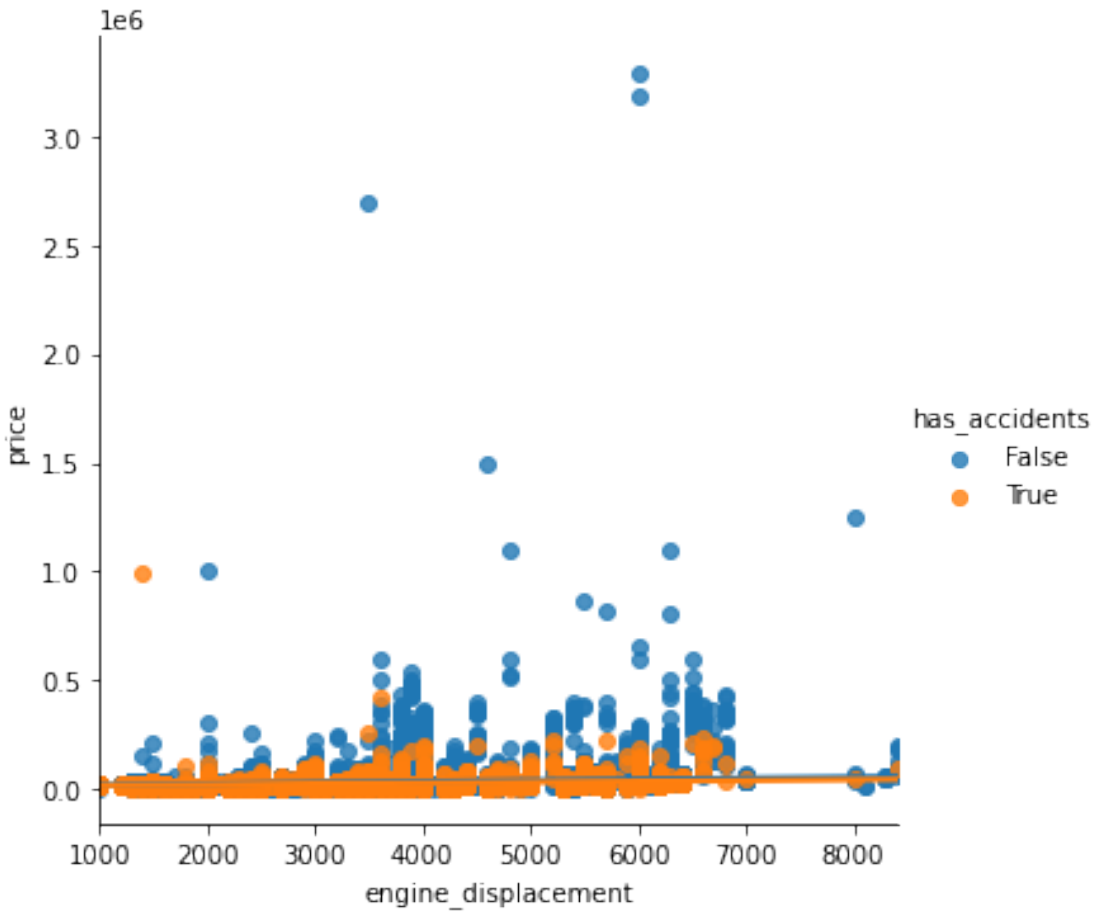


It shows positive relationship between horsepower and price

Visualize the engine_displacement and price with Scatter plot using Seaborn lmplot.

[58]:
```
sns.lmplot(x='engine_displacement', y='price', hue='has_accidents',␣
↪data=df_final)
```

[58]: <seaborn.axisgrid.FacetGrid at 0x7fa1523da580>

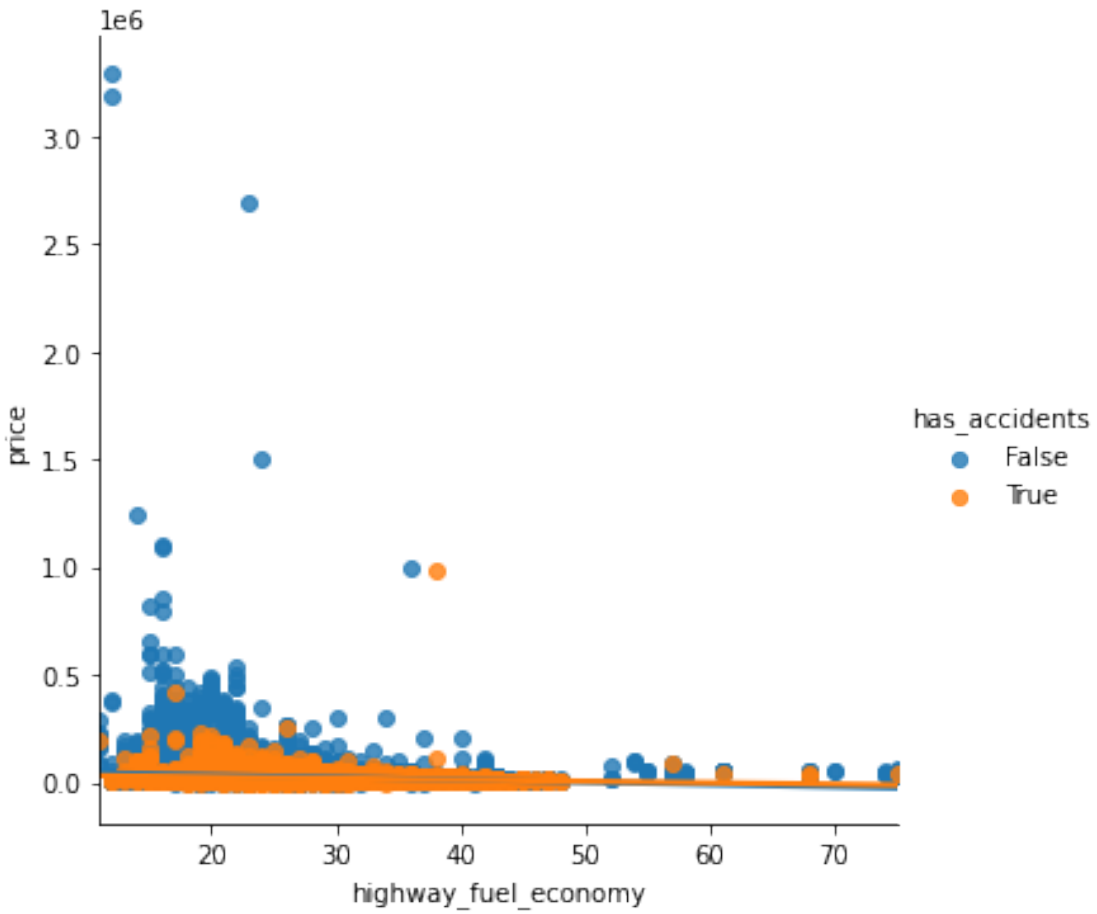


This shows positve relationship as well between price and engine_displacement

Visualize the higway_fuel_economy and price with Scatter plot using Seaborn lmplot.

[55]:
```
sns.lmplot(x='highway_fuel_economy', y='price', hue='has_accidents',␣
↪data=df_final)
```

[55]:
```
<seaborn.axisgrid.FacetGrid at 0x7fa1e53a9220>
```



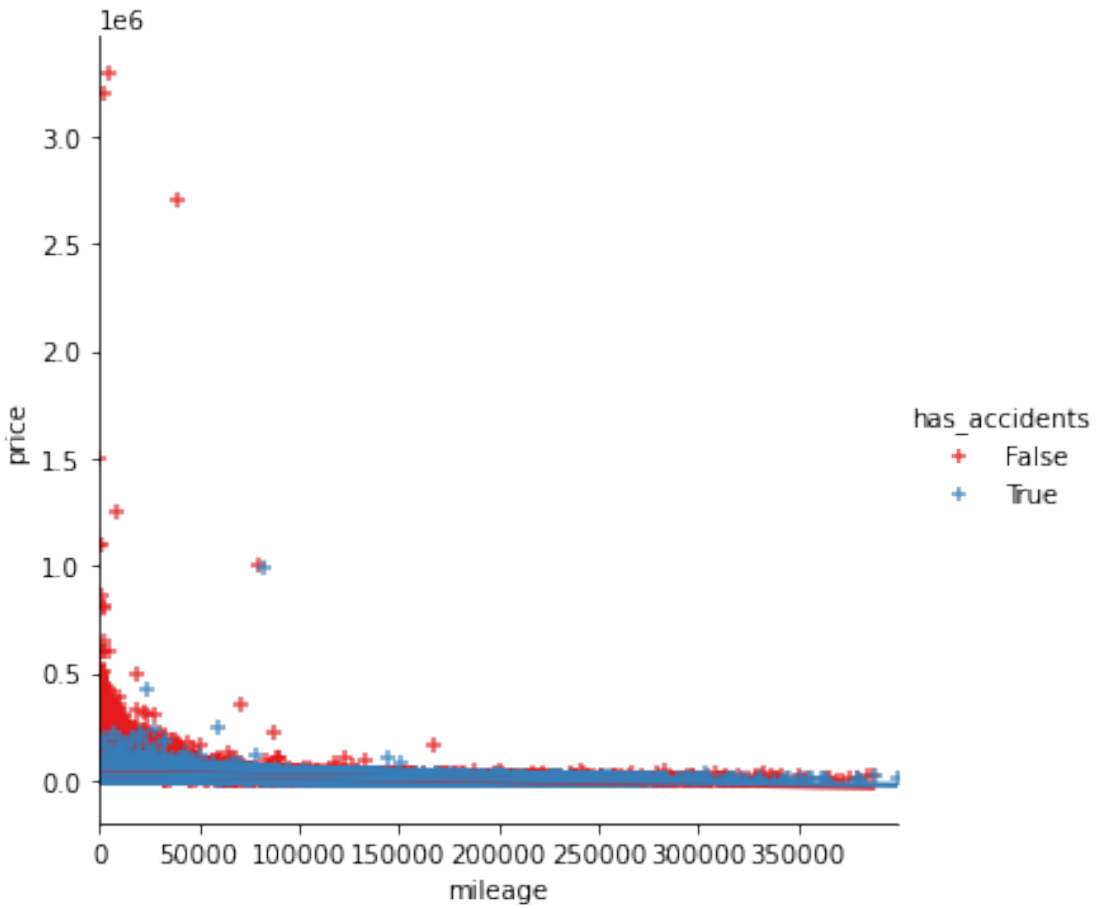
This shows negtive relationship as well between price and highway_fuel_economy

The lower the mileage, the higher the price, and that goes with the general notion. However, the relation breaks when the mileage goes beyond 50k as the price does not vary much from that point.

Visualize the relationship between mileage and price with Scatter plot using Seaborn lmplot.

[62]:
```
sns.lmplot(x='mileage', y='price', hue='has_accidents',palette="Set1",␣
↪markers='+', data=df_final)
```

[62]:
```
<seaborn.axisgrid.FacetGrid at 0x7fa13a283e20>
```



This shows negtive relationship as well between price and mileage

1.0.12 Explore Joint Attributes

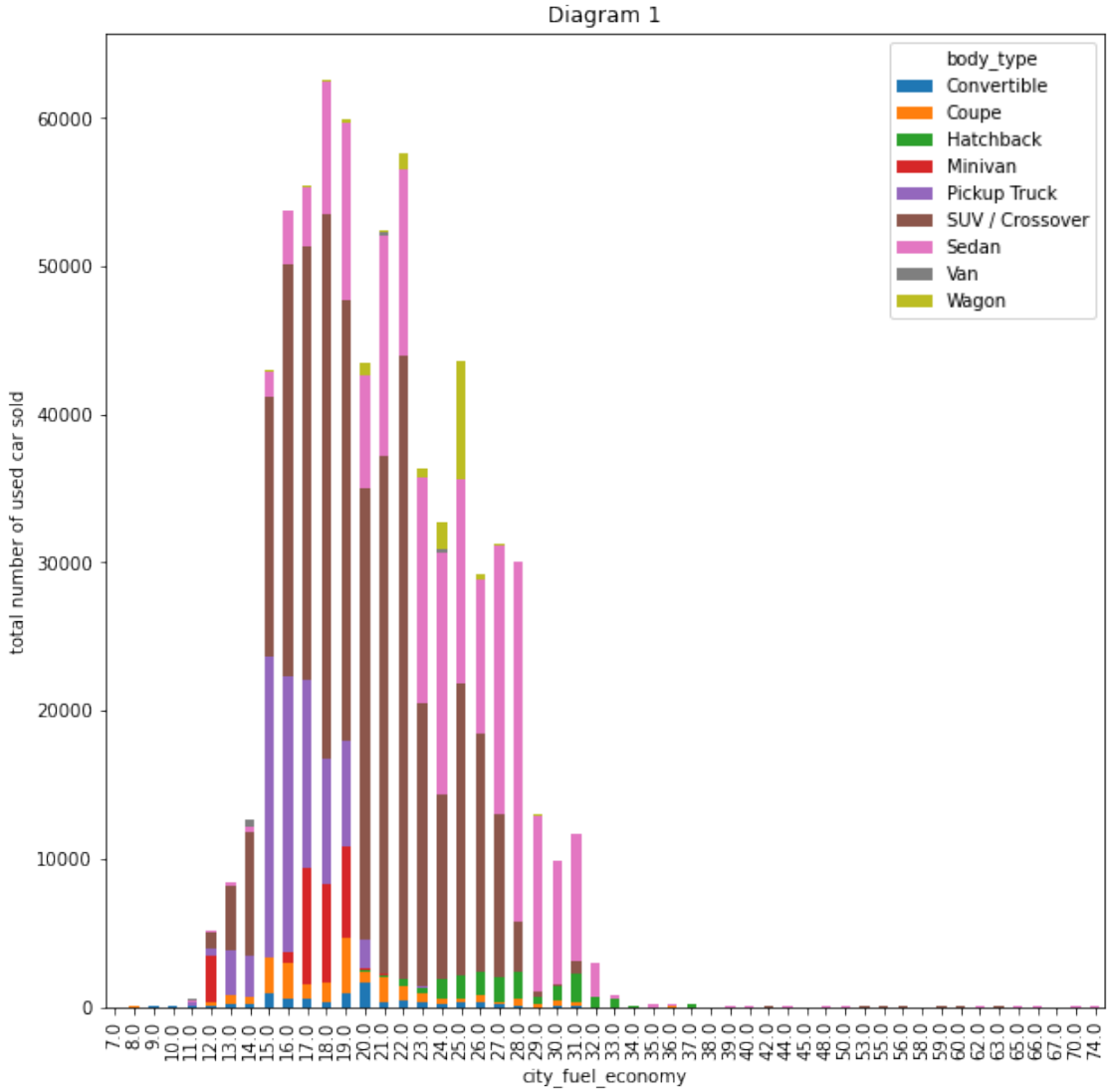
When it comes to joint attributes, we would like to see if there's correlation between "car body type" and "city_fuel_economy".

First, we asked:

*What's the most popular **city fuel economy** for used SUV, pickup Truck, sedan and convertible?* (Please see Diagram 1) * The most popular "city fuel economy" for **used Car** overall is 18 mile * The most popular "city fuel economy" for **used SUV** is 22 mile * The most popular "city fuel economy" for **used Pickup Truck** is 15 mile * The most popular "city fuel economy" for **used Sedan** is 28 mile * The most popular "city fuel economy" for **used convertible** is 20 mile

[61]:
```
df_graph1 = df_final.groupby(['city_fuel_economy', 'body_type'])['body_type'].
 ↪count().unstack().fillna(0)
df_graph1.plot(figsize=(10,10), kind="bar", stacked = True, ylabel = "total␣
 ↪number of used car sold", title = "Diagram 1")
```

[61]:
```
<AxesSubplot:title={'center':'Diagram 1'}, xlabel='city_fuel_economy',
ylabel='total number of used car sold'>
```



Another question we asked was:

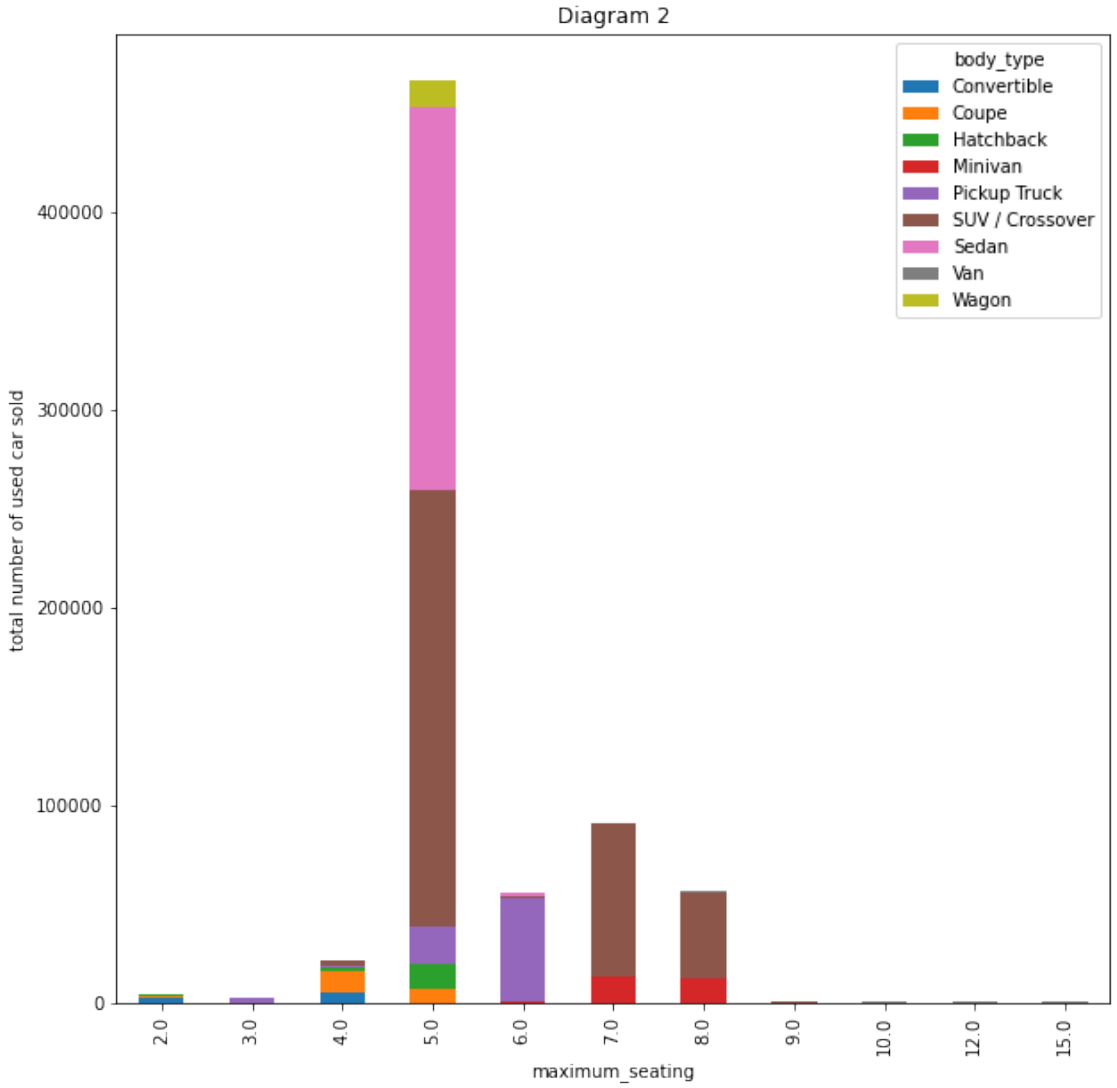
*What's the **highest amount of seating** people are looking for when looking to buy a used car?* (Please see Diagram 2)

- The **most popular** number of seats for used cars is 5 seats
- The **second most popular** number of seats for used car is 6 seats
- When buying a Sedan, customers mostly are looking for 5 seats.

[63]:
```
df_graph2 = df_final.groupby(['maximum_seating', 'body_type'])['body_type'].
↪count().unstack().fillna(0)
```

```
df_graph2.plot(figsize=(10,10), kind="bar", stacked = True, ylabel = "total␣
↪number of used car sold", title = "Diagram 2")
```

[63]: <AxesSubplot:title={'center':'Diagram 2'}, xlabel='maximum_seating', ylabel='total number of used car sold'>



1.0.13 Explore Attributes and Class

we are using price (for regression), has_accidents(for logistic regression) and body_type(for multiclass classifications) as the response variables. In this section, we will try to find some interesting relations involving these features.

'has-accidents' and 'is_new' are two booleans variables in our final dataset. We could have either or these these two values for the binary classification. But has_accidents feature is more balanced

(false: 87% and true:13%) than that of is_new. That is the another driver, besides what is stated in the Business Understanding section, why we leaned on has_accidents for the binary response variable. However, we could have used is_new for the response variable. In that case, we should downscale or upscale, though the downscaling is preferred, training data, which would be an additional step.

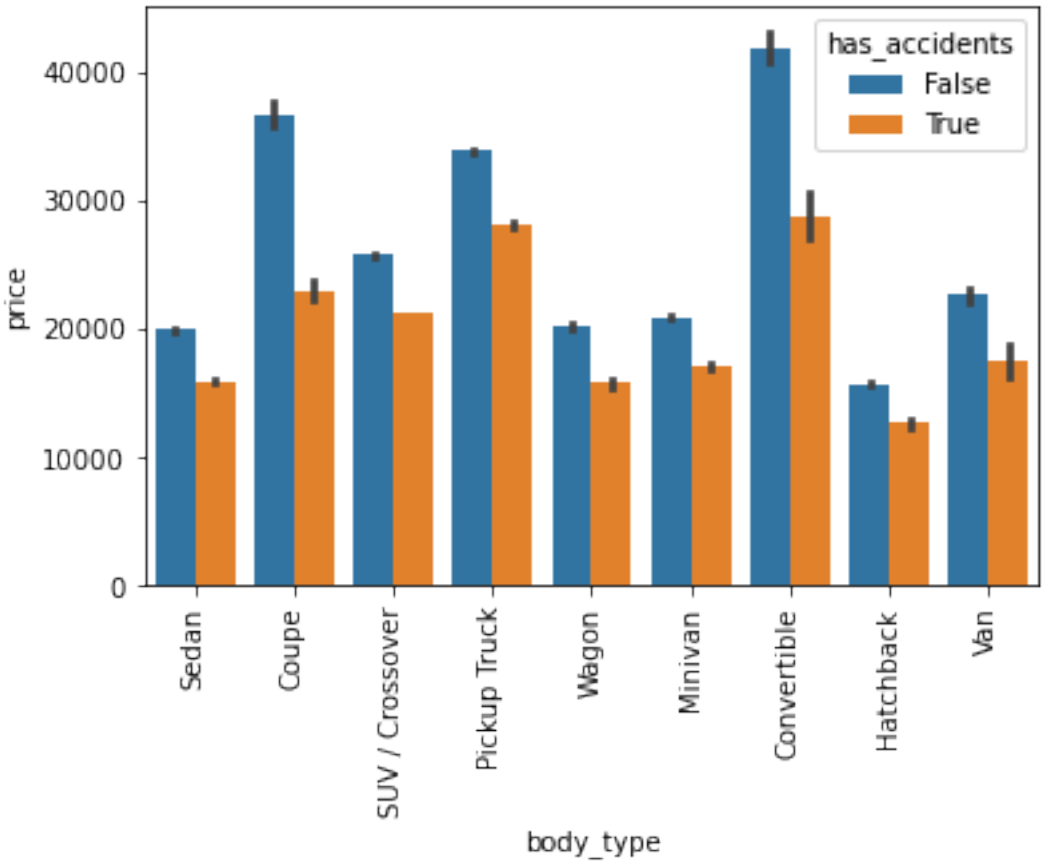
[64]:
```
print('--attribute counts of is_new--')
print(df_final['is_new'].value_counts())

print('--attribute counts of has_accidents--')
print(df_final['has_accidents'].value_counts())
```

```
--attribute counts of is_new--
False    697322
True        667
Name: is_new, dtype: int64
--attribute counts of has_accidents--
False    613283
True      84706
Name: has_accidents, dtype: int64
```

One may think the drivers of Sendan, Minivan or wagon may be more cautious that those who drive SUV or Pickup Truck. But at least based on used car market as presented in this visual, there is not much difference in the driving behavior amoung drives of different body types. However, the drivers of coupe and convertible exhibit restriant while on the road because the damage of their vehicle would cost them dearly.

[65]:
```
Price_bodytype_accidents = sns.barplot(x="body_type", y="price", hue =␣
↪"has_accidents",data=df_final)
Price_bodytype_accidents.set_xticklabels(Price_bodytype_accidents.
↪get_xticklabels(), rotation=90)
```

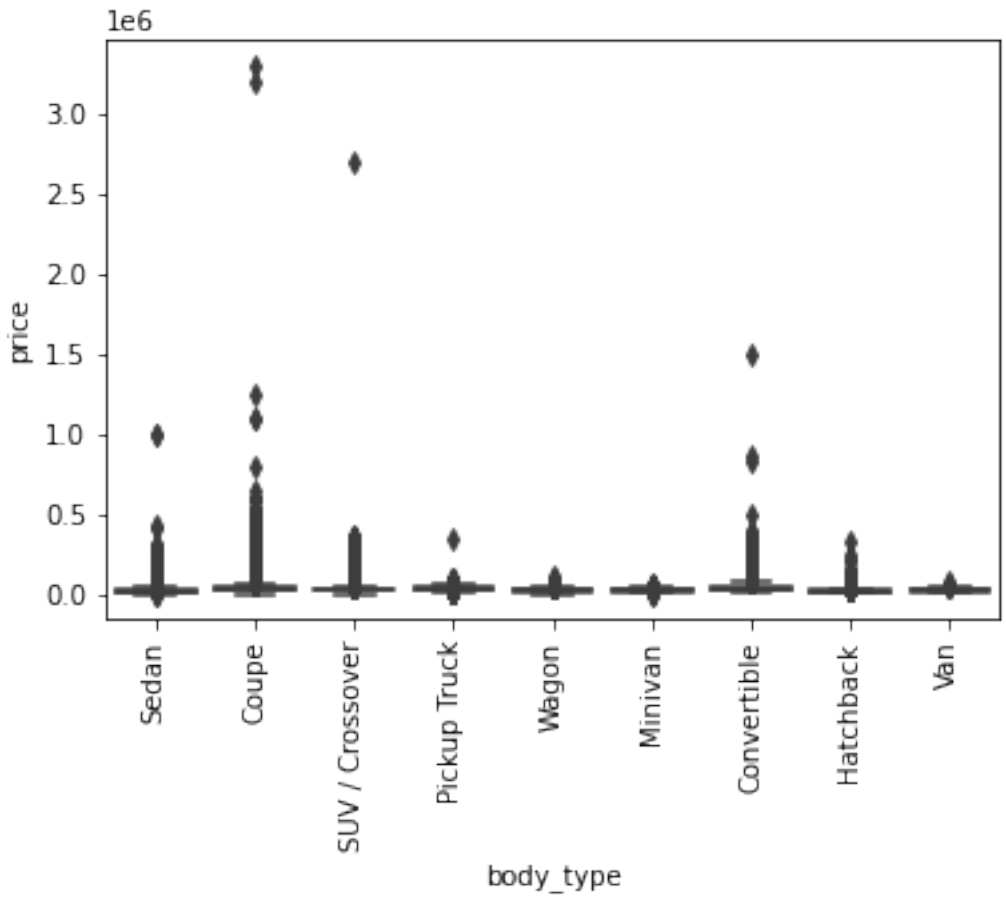
[65]:
```
[Text(0, 0, 'Sedan'),
 Text(1, 0, 'Coupe'),
 Text(2, 0, 'SUV / Crossover'),
 Text(3, 0, 'Pickup Truck'),
 Text(4, 0, 'Wagon'),
 Text(5, 0, 'Minivan'),
 Text(6, 0, 'Convertible'),
 Text(7, 0, 'Hatchback'),
 Text(8, 0, 'Van')]
```



Coupe, SUV/Crossover, Convertible and Sedan have many outliers, whereas Wagon, Minivan and Vad do not have any outlier. In median used car price of 9 body types are with a close band.

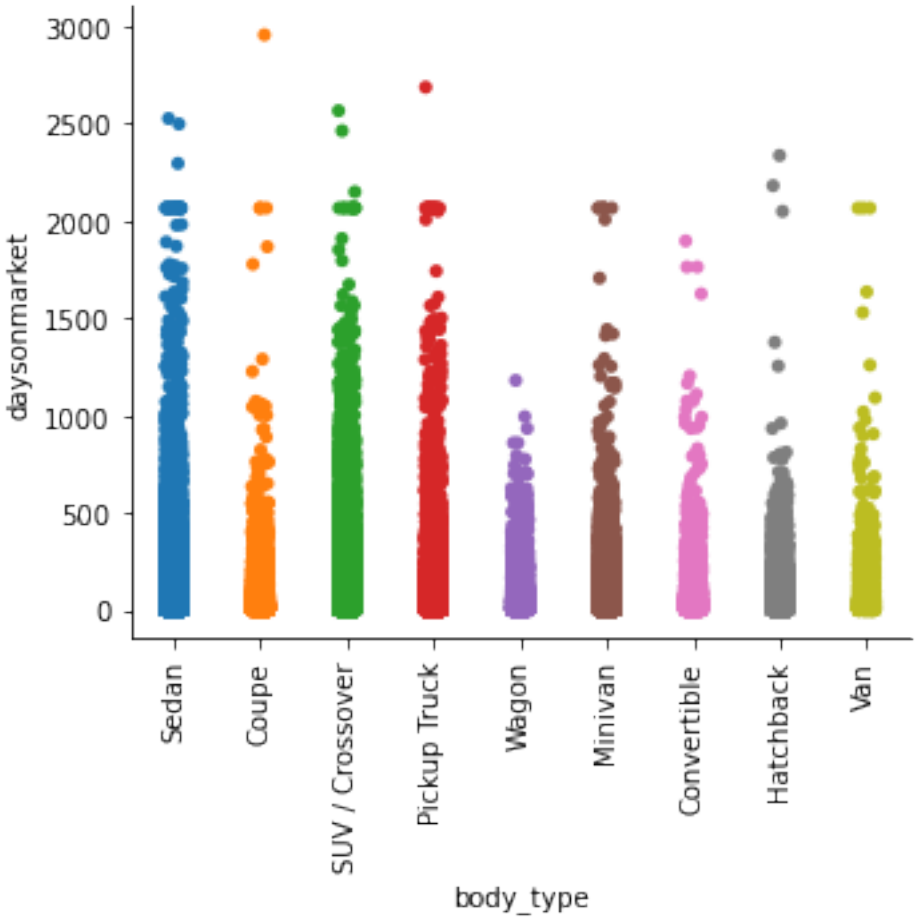
[66]:
```
Price_bodytype = sns.boxplot(x="body_type", y="price", data=df_final)
Price_bodytype.set_xticklabels(Price_bodytype.get_xticklabels(), rotation=90)
```

[66]:
```
[Text(0, 0, 'Sedan'),
 Text(1, 0, 'Coupe'),
 Text(2, 0, 'SUV / Crossover'),
 Text(3, 0, 'Pickup Truck'),
 Text(4, 0, 'Wagon'),
 Text(5, 0, 'Minivan'),
 Text(6, 0, 'Convertible'),
 Text(7, 0, 'Hatchback'),
 Text(8, 0, 'Van')]
```



Sedan, SUV/Crossover, Pickup Truck stay in the market much loger than Coupe, Wagon and Hackback do. It is surprise to see that Coupe, though it is an expensive category car, sells faster Sedan, which we believe a common people's car. Most likely supply demand or mispricing is causing this anamoly.

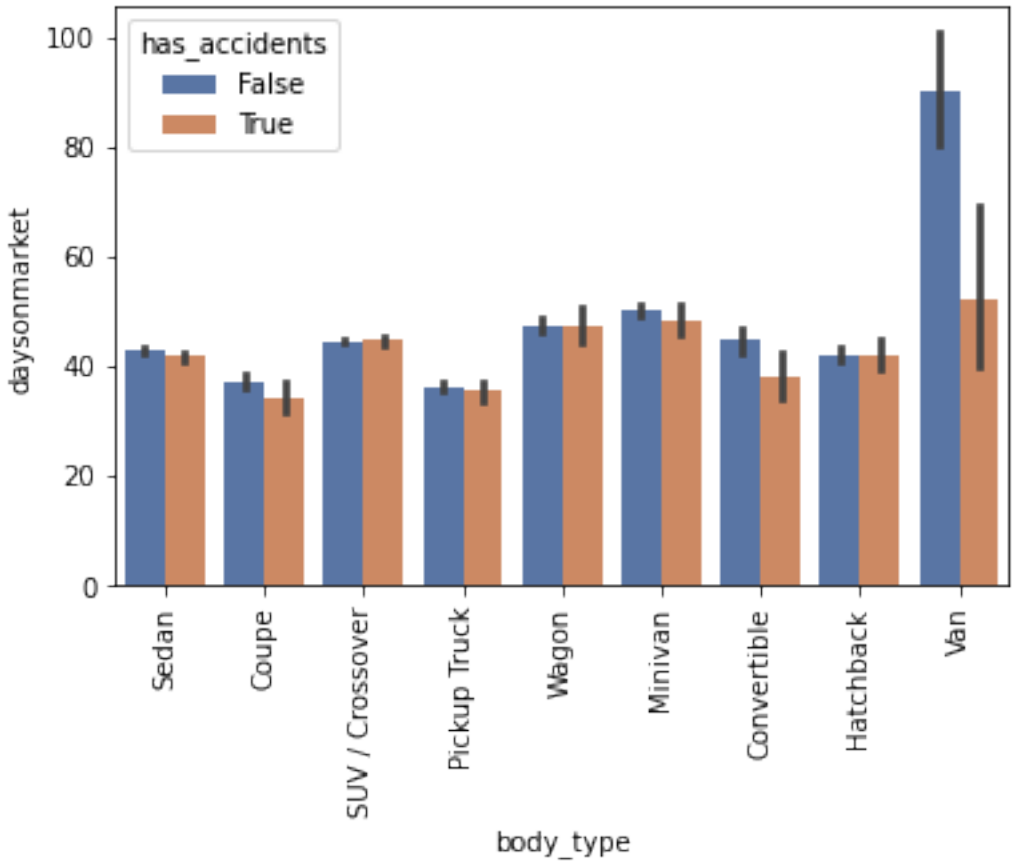
[67]:
```
plot = sns.catplot(x="body_type", y="daysonmarket", data=df_final)
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=90)
plt.tight_layout()
```



We may think that the used cars having accidents will stay longer in the market, but that is not the case with an exception of Van.

[68]:
```
Plt3 = sns.barplot(x="body_type", y="daysonmarket", hue = "has_accidents",␣
↪palette="deep",data=df_final)
Plt3.set_xticklabels(Plt3.get_xticklabels(), rotation=90)
```

[68]:
```
[Text(0, 0, 'Sedan'),
 Text(1, 0, 'Coupe'),
 Text(2, 0, 'SUV / Crossover'),
 Text(3, 0, 'Pickup Truck'),
 Text(4, 0, 'Wagon'),
 Text(5, 0, 'Minivan'),
 Text(6, 0, 'Convertible'),
 Text(7, 0, 'Hatchback'),
 Text(8, 0, 'Van')]
```



1.0.14 New Features

We further group numerical predictors, including "price", to help us to gain more insight.

This new feature is created using the attribute 'price'. We divide price into different price groups as following:

- "<5000" : price < 5000
- "5000-10000" : 5000 <= price <= 10000
- "10000-15000" : 10000 < price <= 15000
- "15000-20000" : 15000 < price <= 20000
- "20000-25000" : 20000 < price <= 25000
- "25000 and over" : price > 25000

[78]:
```
price_group = []
for price in df_final["price"]:
    if price < 5000:
        price_group.append("<5000")
    elif 5000 <= price <= 10000:
        price_group.append("5000-10000")
    elif 10000 < price <= 15000:
```

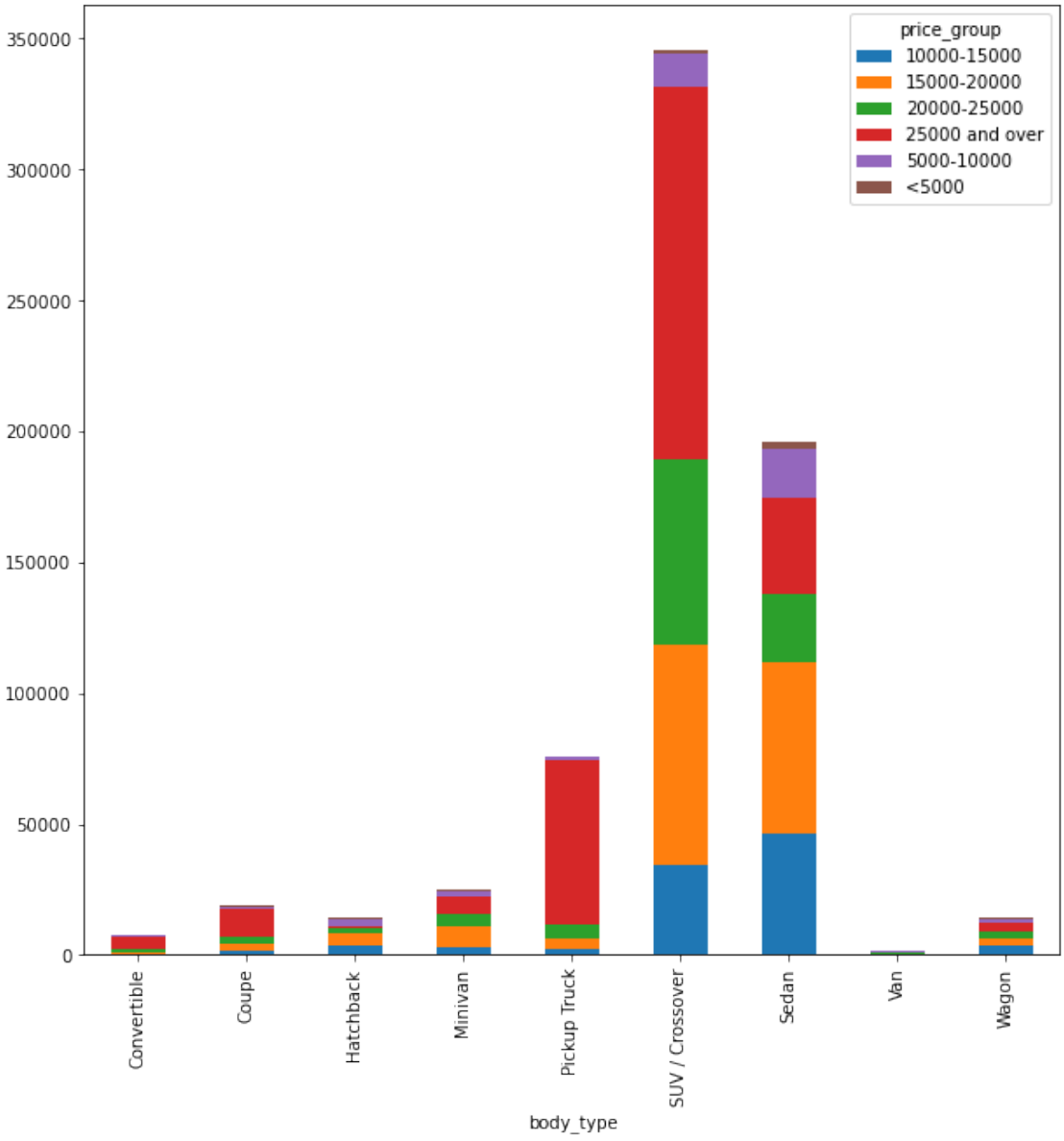
```
        price_group.append("10000-15000")
    elif 15000 < price <= 20000:
        price_group.append("15000-20000")
    elif 20000 < price <= 25000:
        price_group.append("20000-25000")
    else:
        price_group.append("25000 and over")
```

[79]:
```
new_df_price = df_final.copy()
new_df_price["price_group"] = price_group
del new_df_price["price"]
```

Obervations after we had used the new price group for data analysing: * Over 65% of customers spent 25000 and more when buying a used Pickup truck * Over 50% of customers spent over 20000 for a used SUV * Over 50% of customers spent between 10000-20000 when buying a used sedan

[80]:
```
df_price = pd.crosstab(new_df_price.body_type, new_df_price.price_group)
df_price.plot(figsize=(10,10), kind="bar", stacked = True)
```

[80]:
```
<AxesSubplot:xlabel='body_type'>
```



1.0.15 Exceptional Work

In this section, we will use *one hot encoding* to encode the data set below

[81]:
```
import numpy as np
import pandas as pd
import sklearn.preprocessing as preprocessing
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn.model_selection as cross_validation
import sklearn.linear_model as linear_model
```

```
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
```

First, we encode the categorical features as numbers. (see below function)

[82]:
```
def number_encode_features(df_final):
    result = df_final.copy()
    encoders = {}
    for column in result.columns:
        if result.dtypes[column] == np.object:
            encoders[column] = preprocessing.LabelEncoder()
            result[column] = encoders[column].fit_transform(result[column])
    return result, encoders
```

Below shows a snap shot of what the final data looks like after categorical data has been encoded. You can see the body type is in a numerical representation, instead of a string (object) type, before being encoded.

As indicated in the heatmap in the sample statistics (above) section, in the future we want to further explore the strong correlation between "price" and "body_type".

[83]:
```
encoded_data = number_encode_features(df_final)
encoded_data
```

[83]:
```
(         body_type  city_fuel_economy  daysonmarket  engine_displacement  \
 38               6               27.0            55               1500.0
 40               1               18.0            36               3500.0
 41               5               18.0            27               3600.0
 45               5               15.0            27               3600.0
 47               5               18.0            24               3600.0
 …              …                 …               …                  …
 3000026          5               26.0            32               1400.0
 3000028          5               26.0            17               2500.0
 3000031          6               26.0            17               2500.0
 3000034          4               18.0            89               3500.0
 3000039          5               26.0            17               2500.0

          frame_damaged  has_accidents  height  highway_fuel_economy  \
 38                   0              0    57.6                  36.0
 40                   0              0    55.1                  24.0
 41                   0              0    70.7                  27.0
 45                   0              1    69.9                  22.0
 47                   0              0    69.3                  25.0
 …                  …              …      …                     …
 3000026              0              0    66.0                  31.0
 3000028              0              0    66.4                  32.0
 3000031              0              0    57.9                  37.0
```

```
3000034              0              0    70.6                  23.0
3000039              0              0    68.1                  33.0

         horsepower  is_new  length  maximum_seating  mileage  owner_count  \
38            160.0   False    57.6              5.0  42394.0          1.0
40            311.0   False    55.1              4.0  62251.0          1.0
41            310.0   False    70.7              8.0  36410.0          1.0
45            281.0   False    69.9              8.0  36055.0          1.0
47            295.0   False    69.3              5.0  25745.0          1.0
…               …     …      …                 …      …            …
3000026       138.0   False    66.0              5.0   7444.0          1.0
3000028       170.0   False    66.4              5.0  20160.0          1.0
3000031       179.0   False    57.9              5.0  62138.0          1.0
3000034       278.0   False    70.6              5.0  20009.0          1.0
3000039       170.0   False    68.1              7.0  22600.0          1.0

           price  seller_rating  width  year
38       14639.0       3.447761   73.0  2018
40       32000.0       2.800000   81.5  2018
41       23723.0       3.447761   78.6  2018
45       22422.0       3.447761   78.5  2017
47       29424.0       3.447761   84.8  2018
…          …             …      …    …
3000026  17836.0       4.533333   69.9  2019
3000028  20700.0       4.333333   80.0  2017
3000031  17700.0       4.333333   72.0  2018
3000034  40993.0       5.000000   75.2  2017
3000039  19900.0       4.333333   72.4  2017

[697989 rows x 18 columns],
{'body_type': LabelEncoder(),
 'frame_damaged': LabelEncoder(),
 'has_accidents': LabelEncoder()})
```