Exploratory Data Analysis of Used Vehicle

Craigslist is the world's largest collection of used vehicles for sale, this dataset which includes used vehicle entry within the United States.This data contains most all relevant information that Craigslist provides on car sales including columns like price, condition, manufacturer, latitude/longitude, and 18 other categories.Explore the world's largest collection of used vehicles for sale in the US through this Exploratory Data Analysis project. Get insights on price, condition, manufacturer and more.

Data Download

Opendatasets is a Python library for downloading datasets from online sources like Kaggle and Google Drive using a simple Python command.Here I have used opendatasets to download the data from Kaggle using kaggle username and API key.

```
pip install opendatasets --upgrade --quiet
```

```python
import opendatasets as od

download_url = 
'https://www.kaggle.com/datasets/austinreese/craigslist-carstrucks-data'

od.download(download_url)
```

```
Please provide your Kaggle credentials to download this dataset. Learn 
more: http://bit.ly/kaggle-creds
Your Kaggle username: deeksudee
Your Kaggle Key: ··········
Downloading craigslist-carstrucks-data.zip to ./craigslist-carstrucks-
data
```

```
100%|██████████| 262M/262M [00:02<00:00, 106MB/s]
```

```python
data_filename = './craigslist-carstrucks-data/vehicles.csv'
import pandas as pd
df = pd.read_csv(data_filename)
df
```

```
                id                                                url  
\
0       7222695916  https://prescott.craigslist.org/cto/d/prescott...

1       7218891961  https://fayar.craigslist.org/ctd/d/bentonville...

2       7221797935  https://keys.craigslist.org/cto/d/summerland-k...

3       7222270760  https://worcester.craigslist.org/cto/d/west-br...

4       7210384030  https://greensboro.craigslist.org/cto/d/trinit...

...            ...                                                ...

426875  7301591192  https://wyoming.craigslist.org/ctd/d/atlanta-2...

426876  7301591187  https://wyoming.craigslist.org/ctd/d/atlanta-2...

426877  7301591147  https://wyoming.craigslist.org/ctd/d/atlanta-2...

426878  7301591140  https://wyoming.craigslist.org/ctd/d/atlanta-2...

426879  7301591129  https://wyoming.craigslist.org/ctd/d/atlanta-2...


                        region                         region_url  
price  \
0                     prescott    https://prescott.craigslist.org   
6000   
1                 fayetteville       https://fayar.craigslist.org  
11900   
2                 florida keys        https://keys.craigslist.org  
21000   
3       worcester / central MA   https://worcester.craigslist.org   
1500   
4                   greensboro  https://greensboro.craigslist.org   
4900   
...                        ...                                ...    .
```

```
..
426875                 wyoming     https://wyoming.craigslist.org
23590
426876                 wyoming     https://wyoming.craigslist.org
30590
426877                 wyoming     https://wyoming.craigslist.org
34990
426878                 wyoming     https://wyoming.craigslist.org
28990
426879                 wyoming     https://wyoming.craigslist.org
30590

          year manufacturer                     model condition
cylinders  \
0          NaN          NaN                       NaN       NaN
NaN
1          NaN          NaN                       NaN       NaN
NaN
2          NaN          NaN                       NaN       NaN
NaN
3          NaN          NaN                       NaN       NaN
NaN
4          NaN          NaN                       NaN       NaN
NaN
...        ...          ...                       ...       ...
...
426875  2019.0       nissan         maxima s sedan 4d      good  6
cylinders
426876  2020.0        volvo  s60 t5 momentum sedan 4d      good
NaN
426877  2020.0     cadillac          xt4 sport suv 4d      good
NaN
426878  2018.0        lexus           es 350 sedan 4d      good  6
cylinders
426879  2019.0          bmw  4 series 430i gran coupe      good
NaN

        ... size       type paint_color  \
0       ...  NaN        NaN         NaN
1       ...  NaN        NaN         NaN
2       ...  NaN        NaN         NaN
3       ...  NaN        NaN         NaN
4       ...  NaN        NaN         NaN
...     ...  ...        ...         ...
426875  ...  NaN      sedan         NaN
426876  ...  NaN      sedan         red
426877  ...  NaN  hatchback       white
426878  ...  NaN      sedan      silver
426879  ...  NaN      coupe         NaN
```

```
                                                image_url  \
0                                                     NaN   
1                                                     NaN   
2                                                     NaN   
3                                                     NaN   
4                                                     NaN   
...                                                   ...   
426875  https://images.craigslist.org/00o0o_iiraFnHg8q...   
426876  https://images.craigslist.org/00x0x_15sbgnxCIS...   
426877  https://images.craigslist.org/00L0L_farM7bxnxR...   
426878  https://images.craigslist.org/00z0z_bKnIVGLkDT...   
426879  https://images.craigslist.org/00Y0Y_lEUocjyRxa...   

                                              description county state
\
0                                                     NaN    NaN    az

1                                                     NaN    NaN    ar

2                                                     NaN    NaN    fl

3                                                     NaN    NaN    ma

4                                                     NaN    NaN    nc

...                                                   ...    ...   ...

426875  Carvana is the safer way to buy a car During t...    NaN    wy

426876  Carvana is the safer way to buy a car During t...    NaN    wy

426877  Carvana is the safer way to buy a car During t...    NaN    wy

426878  Carvana is the safer way to buy a car During t...    NaN    wy

426879  Carvana is the safer way to buy a car During t...    NaN    wy


              lat       long              posting_date  
0             NaN        NaN                       NaN  
1             NaN        NaN                       NaN  
2             NaN        NaN                       NaN  
3             NaN        NaN                       NaN  
4             NaN        NaN                       NaN  
...           ...        ...                       ...  
426875  33.786500 -84.445400  2021-04-04T03:21:31-0600  
426876  33.786500 -84.445400  2021-04-04T03:21:29-0600  
426877  33.779214 -84.411811  2021-04-04T03:21:17-0600  
426878  33.786500 -84.445400  2021-04-04T03:21:11-0600  
```

```
426879  33.779214 -84.411811  2021-04-04T03:21:07-0600

[426880 rows x 26 columns]
```

Data Preparation and Cleaning

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset.

```
df.columns
```

```
Index(['id', 'url', 'region', 'region_url', 'price', 'year',
'manufacturer',
       'model', 'condition', 'cylinders', 'fuel', 'odometer',
'title_status',
       'transmission', 'VIN', 'drive', 'size', 'type', 'paint_color',
       'image_url', 'description', 'county', 'state', 'lat', 'long',
       'posting_date'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 426880 entries, 0 to 426879
Data columns (total 26 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   id            426880 non-null  int64
 1   url           426880 non-null  object
 2   region        426880 non-null  object
 3   region_url    426880 non-null  object
 4   price         426880 non-null  int64
 5   year          425675 non-null  float64
 6   manufacturer  409234 non-null  object
 7   model         421603 non-null  object
 8   condition     252776 non-null  object
 9   cylinders     249202 non-null  object
 10  fuel          423867 non-null  object
 11  odometer      422480 non-null  float64
 12  title_status  418638 non-null  object
 13  transmission  424324 non-null  object
 14  VIN           265838 non-null  object
 15  drive         296313 non-null  object
 16  size          120519 non-null  object
 17  type          334022 non-null  object
 18  paint_color   296677 non-null  object
 19  image_url     426812 non-null  object
 20  description   426810 non-null  object
 21  county        0 non-null       float64
 22  state         426880 non-null  object
 23  lat           420331 non-null  float64
 24  long          420331 non-null  float64
 25  posting_date  426812 non-null  object 
dtypes: float64(5), int64(2), object(19)
memory usage: 84.7+ MB
```

```
df.describe()
```

```
                 id         price           year      odometer  county  
\
count  4.268800e+05  4.268800e+05  425675.000000  4.224800e+05     0.0   

mean   7.311487e+09  7.519903e+04    2011.235191  9.804333e+04     NaN   

std    4.473170e+06  1.218228e+07       9.452120  2.138815e+05     NaN   

min    7.207408e+09  0.000000e+00    1900.000000  0.000000e+00     NaN   

25%    7.308143e+09  5.900000e+03    2008.000000  3.770400e+04     NaN   

50%    7.312621e+09  1.395000e+04    2013.000000  8.554800e+04     NaN   

75%    7.315254e+09  2.648575e+04    2017.000000  1.335425e+05     NaN   

max    7.317101e+09  3.736929e+09    2022.000000  1.000000e+07     NaN   


                 lat           long  
count  420331.000000  420331.000000  
mean       38.493940     -94.748599  
std         5.841533      18.365462  
min       -84.122245    -159.827728  
25%        34.601900    -111.939847  
50%        39.150100     -88.432600  
75%        42.398900     -80.832039  
max        82.390818     173.885502  
```

```
numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 
'float64']

numerics_df = df.select_dtypes(include=numerics)
len(numerics_df.columns)
```

```
7
```

Percentage of missing values

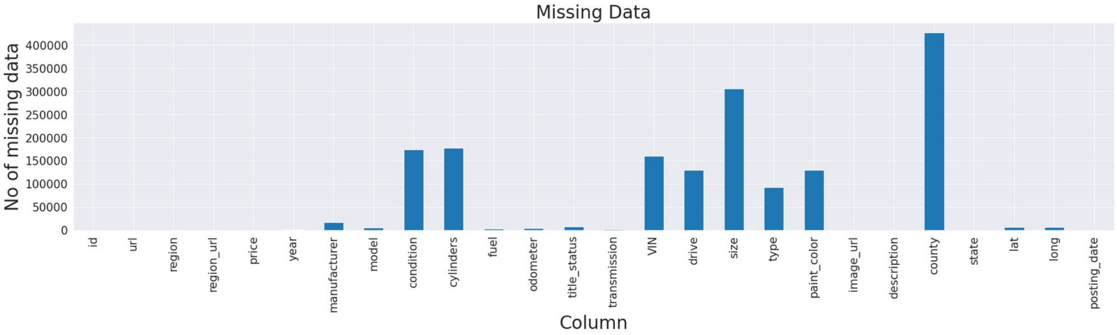
```
missing_percentages = 
df.isna().sum().sort_values(ascending=False)/len(df)
missing_percentages
```

```
county          1.000000
size            0.717675
cylinders       0.416225
condition       0.407852
VIN             0.377254
drive           0.305863
paint_color     0.305011
type            0.217527
manufacturer    0.041337
title_status    0.019308
lat             0.015342
long            0.015342
model           0.012362
odometer        0.010307
fuel            0.007058
transmission    0.005988
year            0.002823
description     0.000164
image_url       0.000159
posting_date    0.000159
url             0.000000
price           0.000000
state           0.000000
region_url      0.000000
region          0.000000
id              0.000000
dtype: float64
```

```python
import matplotlib.pyplot as plt
```

```python
import seaborn as sns
sns.set_style('darkgrid')
```

```python
md = df.isnull().sum().plot.bar(title = 'Missing Data')
md.set_xlabel('Column',fontsize = 24)
md.set_ylabel('No of missing data',fontsize = 24)
plt.rcParams['figure.figsize']=(25,7)
plt.xticks(fontsize = 15)
plt.yticks(fontsize = 15)
md.title.set_size(24)
plt.show()
```



Exploratory Analysis and Visualization

Column that have analysed

- state
- year
- color
- vechile tpe
- cylinders

```
df.columns
```

```
Index(['id', 'url', 'region', 'region_url', 'price', 'year',
'manufacturer',
       'model', 'condition', 'cylinders', 'fuel', 'odometer',
'title_status',
       'transmission', 'VIN', 'drive', 'size', 'type', 'paint_color',
       'image_url', 'description', 'county', 'state', 'lat', 'long',
       'posting_date'],
      dtype='object')
```

STATE

```
a = df.state.unique()
len(a)
```

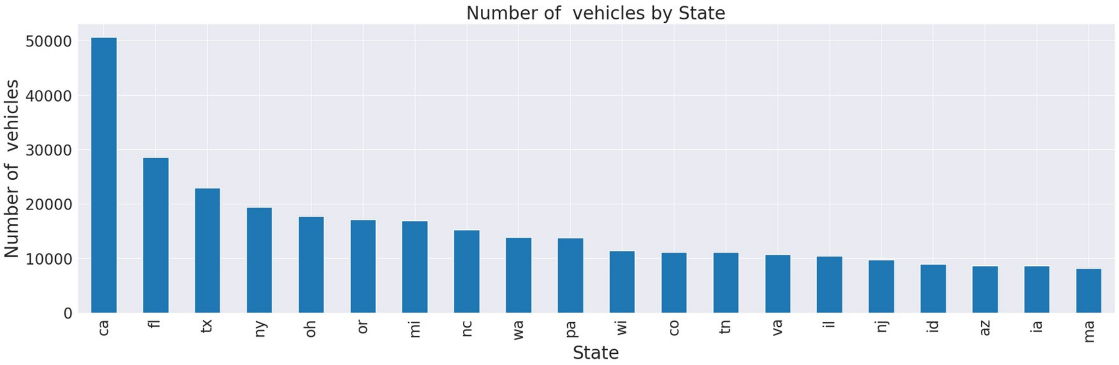
```
51
```

```
vehicle_by_states = df.state.value_counts()
vehicle_by_states
```

```
ca    50614
fl    28511
tx    22945
ny    19386
oh    17696
or    17104
mi    16900
nc    15277
wa    13861
pa    13753
```

```
wi    11398
co    11088
tn    11066
va    10732
il    10387
nj     9742
id     8961
az     8679
ia     8632
ma     8174
mn     7716
ga     7003
ok     6792
sc     6327
mt     6294
ks     6209
in     5704
ct     5188
al     4955
md     4778
nm     4425
mo     4293
ky     4149
ar     4038
ak     3474
la     3196
nv     3194
nh     2981
dc     2970
me     2966
hi     2964
vt     2513
ri     2320
sd     1302
ut     1150
wv     1052
ne     1036
ms     1016
de      949
wy      610
nd      410
Name: state, dtype: int64
```

```
ax =  vehicle_by_states[:20].plot.bar( title='Number of  vehicles by
State', fontsize = 20)
ax.set_xlabel("State", fontsize = 24)
ax.set_ylabel("Number of  vehicles", fontsize = 24)
plt.rcParams['figure.figsize']=(25,7)
ax.title.set_size(24)
```



In terms of volume of sales we can see that California(ca) and Florida(fl) lead the chart.

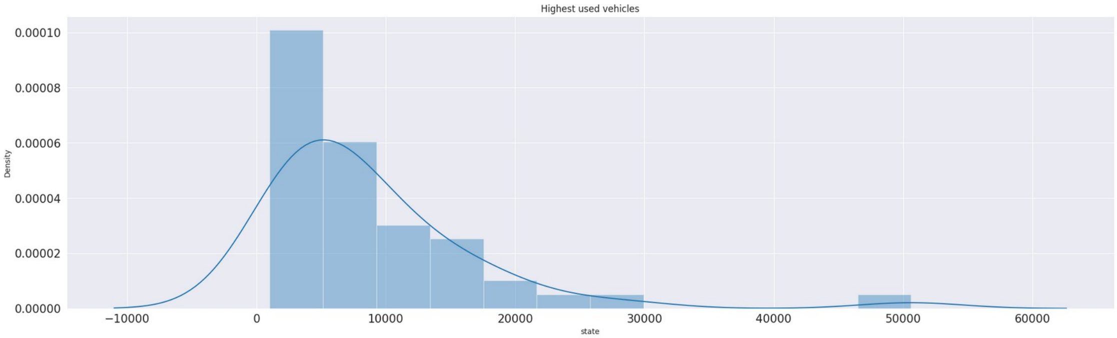
```
high_used_vehicle_states = vehicle_by_states[vehicle_by_states >= 
1000]
low_used_vehicle_states =  vehicle_by_states[vehicle_by_states < 1000]
```

```
len(high_used_vehicle_states)/len(df.state)
```

```
0.00011244377811094453
```

```
sns.distplot(high_used_vehicle_states).set(
     title='Highest used vehicles')
plt.xticks(fontsize = 15)
plt.yticks(fontsize = 15)
plt.rcParams['figure.figsize']=(25,8)
```

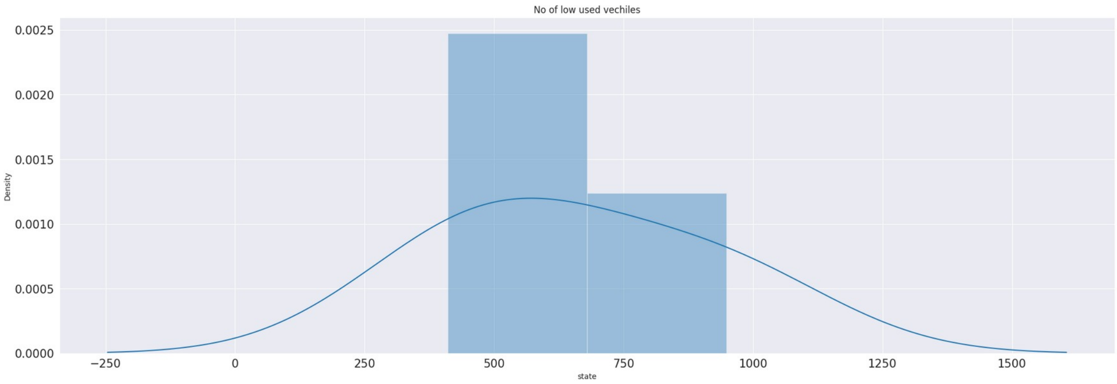
```
<ipython-input-50-3004d575174a>:1: UserWarning: 

`distplot` is a deprecated function and will be removed in seaborn 
v0.14.0.

Please adapt your code to use either `displot` (a figure-level 
function with
similar flexibility) or `histplot` (an axes-level function for 
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(high_used_vehicle_states).set(
```



I have defined high_used_vechiles_states with greater than 1000 vechiles.The above graph shows the highest range of the used vechicles according to state i.e the highest range is 1000 to 10000.

```
sns.distplot(low_used_vehicle_states).set(
     title='No of low used vechiles')
plt.xticks(fontsize = 15)
plt.yticks(fontsize = 15)
```

```
<ipython-input-29-c5a09bcd185a>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(low_used_vehicle_states).set(

(array([0.    , 0.0005, 0.001 , 0.0015, 0.002 , 0.0025, 0.003 ]),
 [Text(0, 0.0, '0.0000'),
  Text(0, 0.0005, '0.0005'),
  Text(0, 0.001, '0.0010'),
  Text(0, 0.0015, '0.0015'),
  Text(0, 0.002, '0.0020'),
  Text(0, 0.0025, '0.0025'),
  Text(0, 0.003, '0.0030')])
```



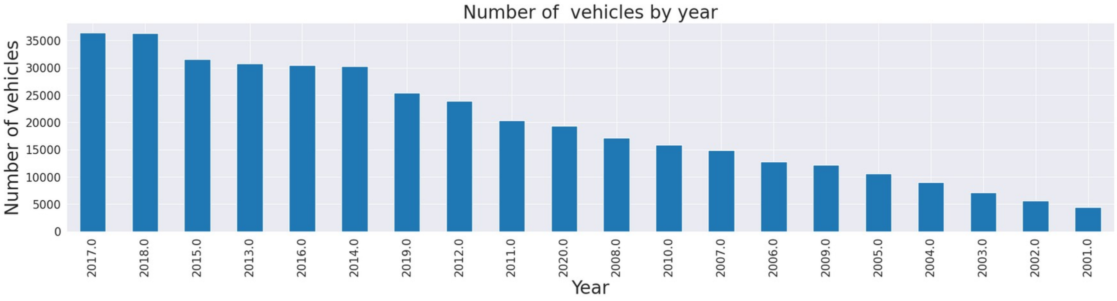
Simlar to highest_used_vechiles_states I have defined low_used_vehicle_states which has vechiles lesser than 1000.The graph shows the less range of vechiles according to state i.e from 500 to 750.

YEAR

```
b = df.price.unique()
len(b)
```

```
15655
```

```
ay = df['year'].value_counts().head(20).plot.bar( title='Number of vehicles by year')
ay.set_xlabel("Year", fontsize = 24)
ay.set_ylabel("Number of vehicles", fontsize = 24)
plt.rcParams['figure.figsize']=(25,5)
plt.xticks(fontsize = 15)
plt.yticks(fontsize = 15)
ay.title.set_size(24)
```



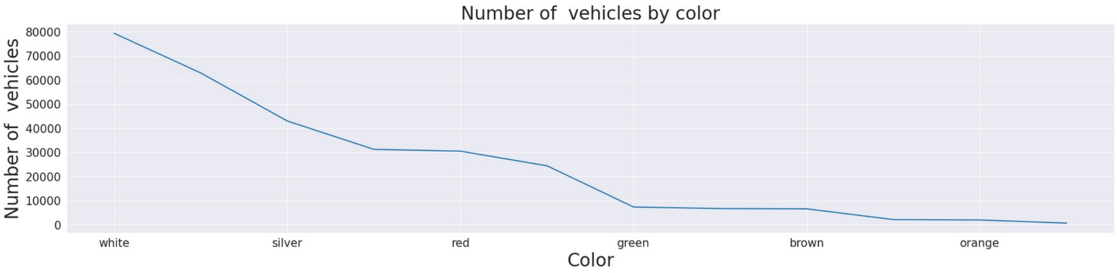
The above garph shows that more no. of vehicles are of year 2017 and 2018 and least is 2001.

COLOR

```
df.paint_color.unique()
```

```
array([nan, 'white', 'blue', 'red', 'black', 'silver', 'grey',
'brown',
       'yellow', 'orange', 'green', 'custom', 'purple'], dtype=object)
```

```
az = df['paint_color'].value_counts().head(20).plot (title='Number of 
vehicles by color')
az.set_xlabel("Color", fontsize = 24)
az.set_ylabel("Number of  vehicles", fontsize = 24)
plt.rcParams['figure.figsize']=(25,5)
plt.xticks(fontsize = 15)
plt.yticks(fontsize = 15)
az.title.set_size(24)
```



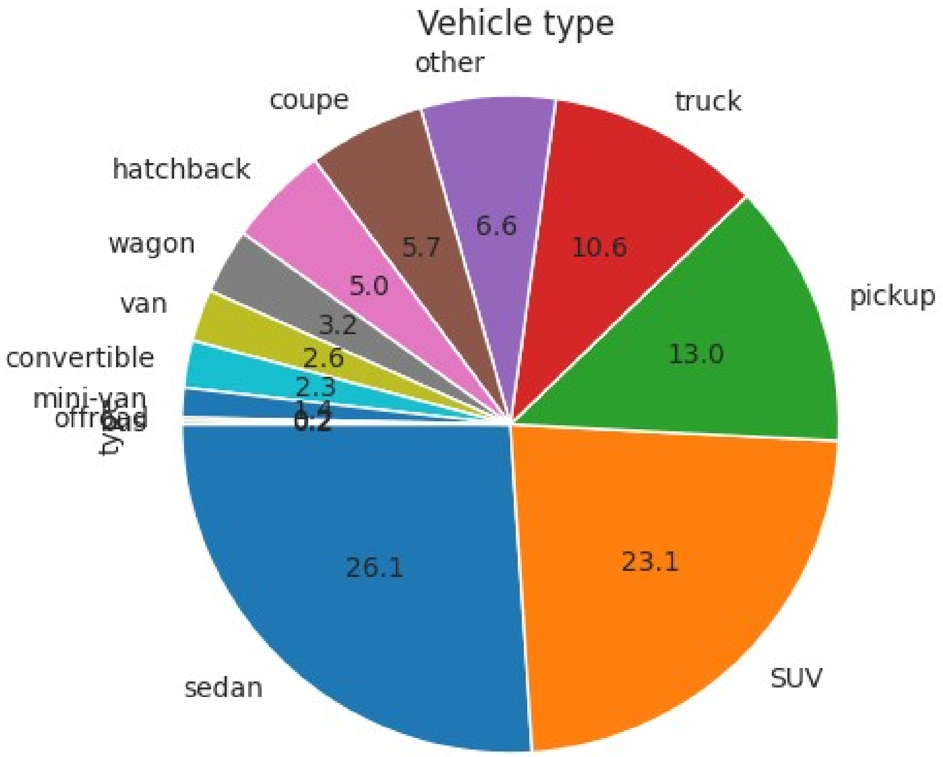
White’s popularity can be attributed to it being one of the easiest colors to maintain, and because it is a common color for fleet and rental vehicles, white is prevalent in the used car market in usa.Similarly garph shows that more no. of vechiles that are for sale are of white color.

VEHICLE TYPE

```
df.type.unique()
```

```
array([nan, 'pickup', 'truck', 'other', 'coupe', 'SUV', 'hatchback',
       'mini-van', 'sedan', 'offroad', 'bus', 'van', 'convertible',
       'wagon'], dtype=object)
```

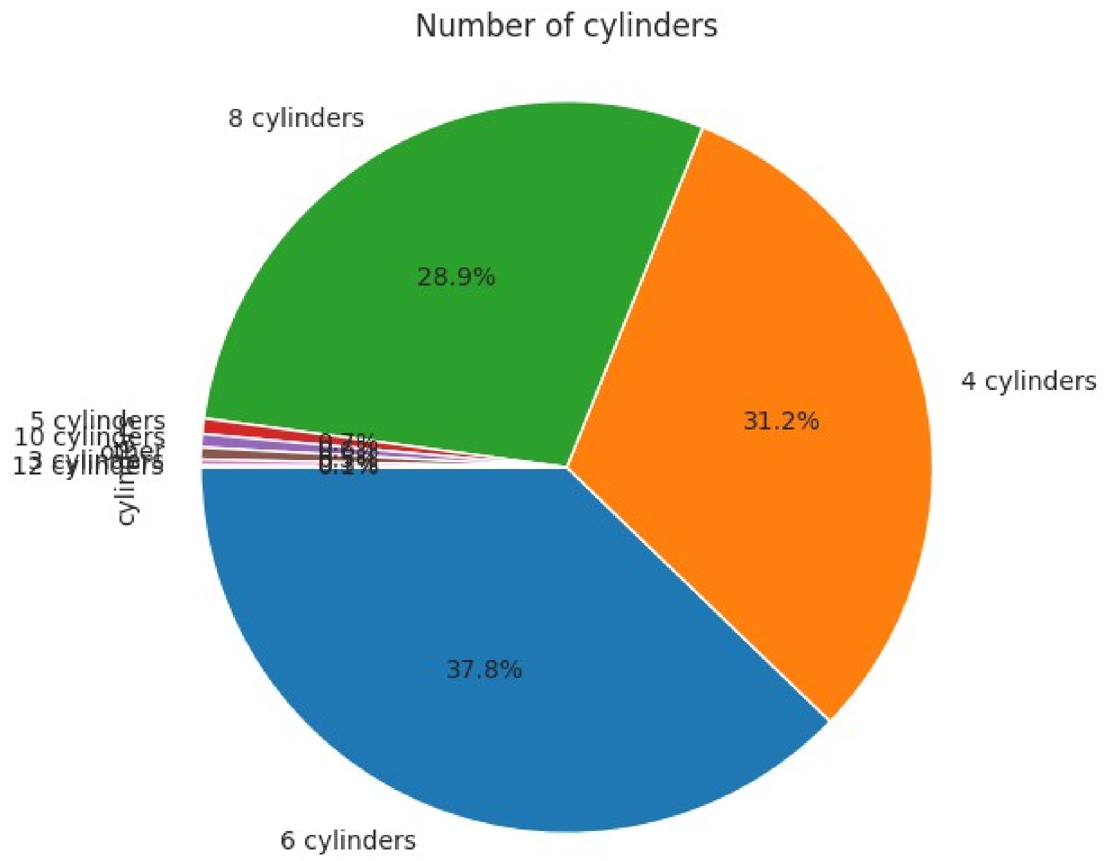
```
df['type'].value_counts().head(20).plot(kind = 'pie',autopct='%1.1f', 
radius = 1.1, startangle=180,title=' Vehicle type');
```



The above graph shows that most of the vechiles for sale are sedan followed by SUV type.

CYLINDER

```
 plt.figure(figsize=(8,6))
 df['cylinders'].value_counts().head(20).plot(kind =
'pie',autopct='%1.1f%%',radius = 1.1,startangle=180, title = 'Number
of cylinders');
```



A six-cylinder engine has more power, but it also uses more gas.According to dataset most of the vechiles has 6 cylinders.

Ask and Answer Questions

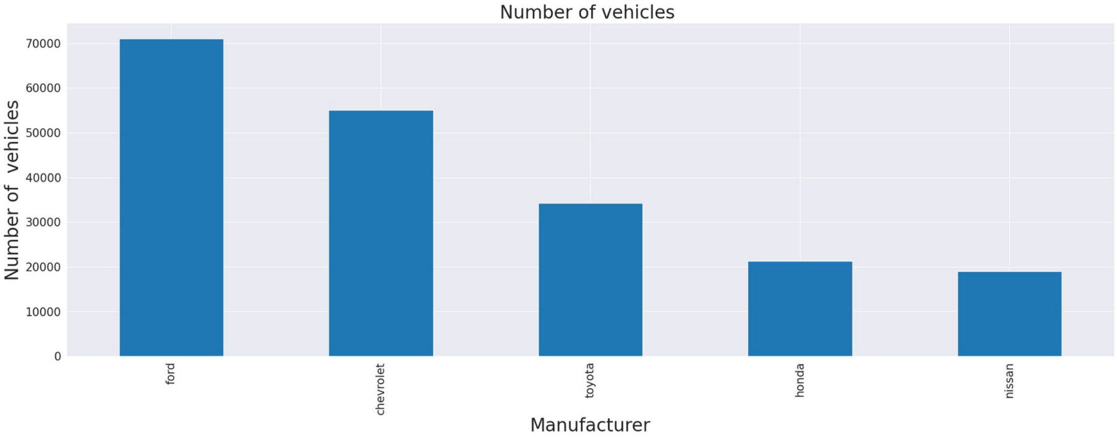
Q1. Which are the top five manufacturing company has highest used cars?

```
 df.manufacturer.unique()
```

```
array([nan, 'gmc', 'chevrolet', 'toyota', 'ford', 'jeep', 'nissan',
'ram',
       'mazda', 'cadillac', 'honda', 'dodge', 'lexus', 'jaguar',
'buick',
       'chrysler', 'volvo', 'audi', 'infiniti', 'lincoln', 'alfa-
romeo',
       'subaru', 'acura', 'hyundai', 'mercedes-benz', 'bmw',
'mitsubishi',
       'volkswagen', 'porsche', 'kia', 'rover', 'ferrari', 'mini',
       'pontiac', 'fiat', 'tesla', 'saturn', 'mercury', 'harley-
davidson',
       'datsun', 'aston-martin', 'land rover', 'morgan'],
dtype=object)
```

```
a = df.manufacturer.value_counts()
```

```
f = a[:5].plot.bar( title='Number of vehicles ')
f.set_xlabel("Manufacturer", fontsize = 24)
f.set_ylabel("Number of  vehicles", fontsize = 24)
plt.rcParams['figure.figsize']=(25,8)
plt.xticks(fontsize = 15)
plt.yticks(fontsize = 15)
f.title.set_size(24)
```



Ford leads the pack with most cars up for sale from a manufacturer. This comes as no surprise since Ford's F-Series line of pickups have been America's Best Selling Truck for 43 years straight. For the past 38 years, Ford F-Series trucks have also been the Best Selling Vehicle in America.Then comes Chevrolet, toyota, honda and nissan.

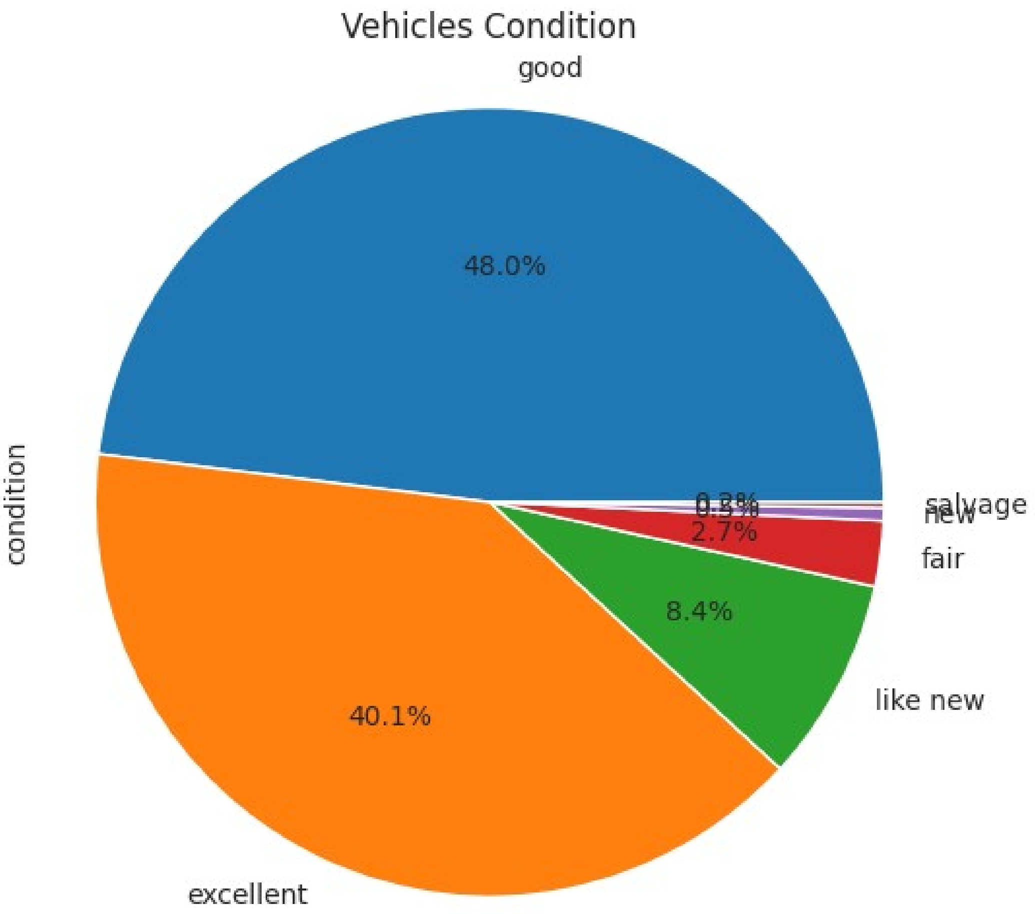
Q2. What is the condition of used vechile for sale?

```
import matplotlib
import matplotlib.pyplot as plt
```

```
df.condition
```

```
0           NaN
1           NaN
2           NaN
3           NaN
4           NaN
          ...
426875    good
426876    good
426877    good
426878    good
426879    good
Name: condition, Length: 426880, dtype: object
```

```
plt.figure(figsize=(8,6))
df['condition'].value_counts().plot(kind = 'pie',radius = 1.1,
autopct='%1.1f%%', title = 'Vehicles Condition');
```



It seems like most of the vechiles are maintained in good condition. This means the vehicle has some repairable cosmetic defects and is free of major mechanical problems.

Q3. What is the percentage of newly purchased vechiles are for sale?

```
total_cars = len(df.condition)
total_cars

426880

new_cars = df.condition.value_counts().new
new_cars

1305

percentage_of_new_cars = (new_cars/total_cars) * 100
percentage_of_new_cars

0.30570652173913043
```

Out of all the vehicles there is only 0.3% of newly purchased vehicles which are for sale.

Q4. Is there any trend in the price over year?

```
import plotly.express as px
import matplotlib.pyplot as plt

fig = px.scatter(df,
            x='year',
            y='price',
            log_y = True)


fig.update_layout(

            title=dict(
            text='Price vs. Year',
      font=dict(
            family="Arial",
            size=24

        )
    ),
    xaxis_title="Year",
    yaxis_title="Price",
    font=dict(
        family="Arial",
        size=24

    )
)
fig.show()
```

In general price and year of the vehicles are inversly proportional but according to this dataset there is no much difference in price over year.

Q5. What is the reading of odometer over price?

```
fig = px.scatter(df,
                 x='odometer',
                 y='price',
                opacity=0.8,
                 hover_data=['year'],
                 title='Price vs. Odometer')
fig.update_traces(marker_size=5)
fig.update_layout(
            height=400,
            title=dict(
            text='Price vs. Odometer',
      font=dict(
            family="Arial",
            size=24
```

```
        )
    ),
    xaxis_title="Odometer",
    yaxis_title="Price",
    font=dict(
        family="Arial",
        size=24
    )
)

fig.show()
```

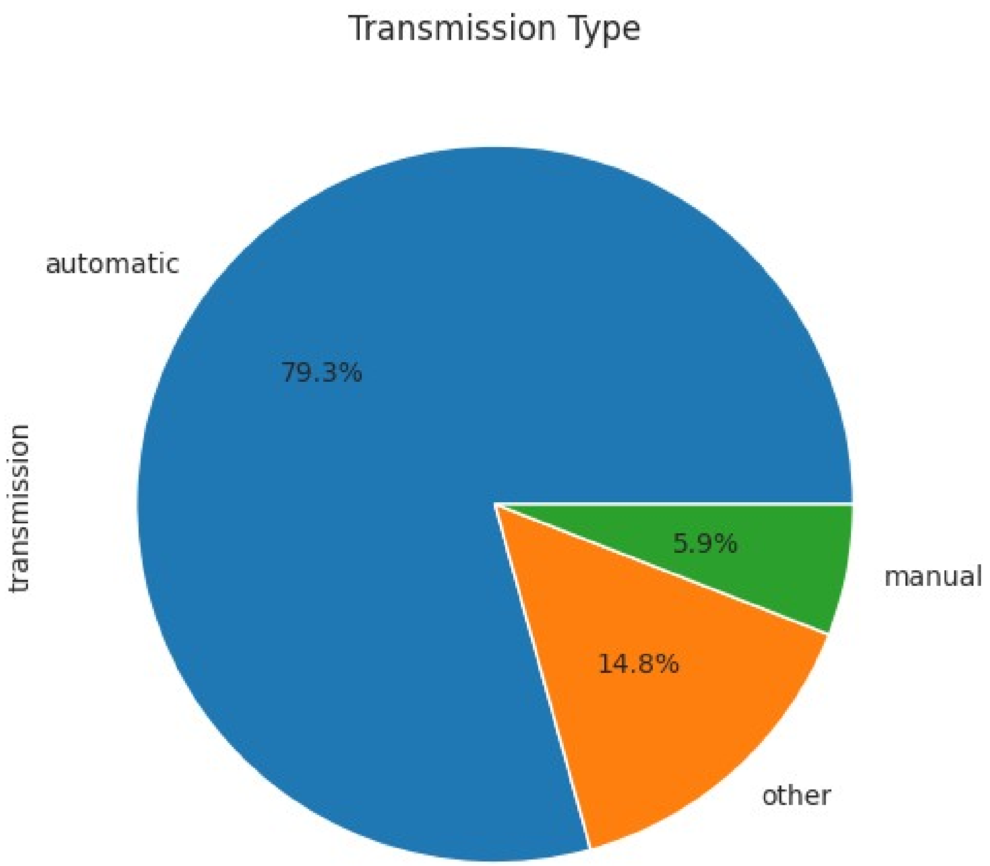
In general price and odometer are inversly proportional if the price is high than odometer reading is low and viseversa, but in this dataset there is no much difference in price for vechiles which have less odometer.Only few vechiles are expensive with less odometer value.

Q6. What is the transmission rate?

```
df.transmission.unique()
```

```
array([nan, 'other', 'automatic', 'manual'], dtype=object)
```

```
plt.figure(figsize=(8,6))
df['transmission'].value_counts().plot(kind = 'pie', autopct='%1.1f%
%', title='Transmission Type');
```



According to CarMax, 96 percent of Americans drive automatics. And, unsurprisingly given that statistic, people just aren't buying cars with manual transmissions in the United States and graph also shows that there are more number of automatic vehicles for sale.

Q7. What is price of used vechiles according automotive industry?

```
df.drive.unique()
```

```
array([nan, 'rwd', '4wd', 'fwd'], dtype=object)
```

```
fig = px.bar(df,
            x='manufacturer',
            y='price',
            hover_data=['drive'],
            title='Price vs. Manufacturer')
fig.update_layout(
            height=400,
            title=dict(
            text='Price vs. Manufacturer',
      font=dict(
            family="Arial",
            size=24
```

```
        )
    ),
    xaxis_title=" Manufacturer",
    yaxis_title="Price",
    font=dict(
        family="Arial",
        size=24
    )
    )

fig.show()
```

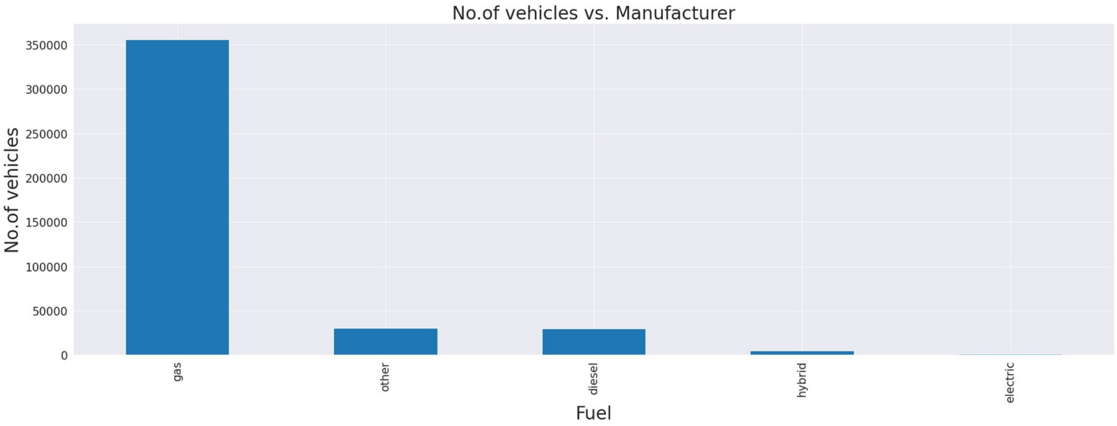
Generally Toyato vechiles are less expensive than ford but according to this dataset the used vechiles of toyato is more costly.

Q8. What is type of fuel uesd by vehicles over year and which fuel type is used more?

```
df.fuel.unique()
```

```
array([nan, 'gas', 'other', 'diesel', 'hybrid', 'electric'],
dtype=object)
```

```
f = df['fuel'].value_counts().plot(kind = 'bar', title='No.of vehicles
vs. Manufacturer')
f.set_xlabel('Fuel', fontsize = 24)
f.set_ylabel('No.of vehicles', fontsize = 24)
plt.rcParams['figure.figsize']=(25,5)
plt.xticks(fontsize = 15)
plt.yticks(fontsize = 15)
f.title.set_size(24);
```



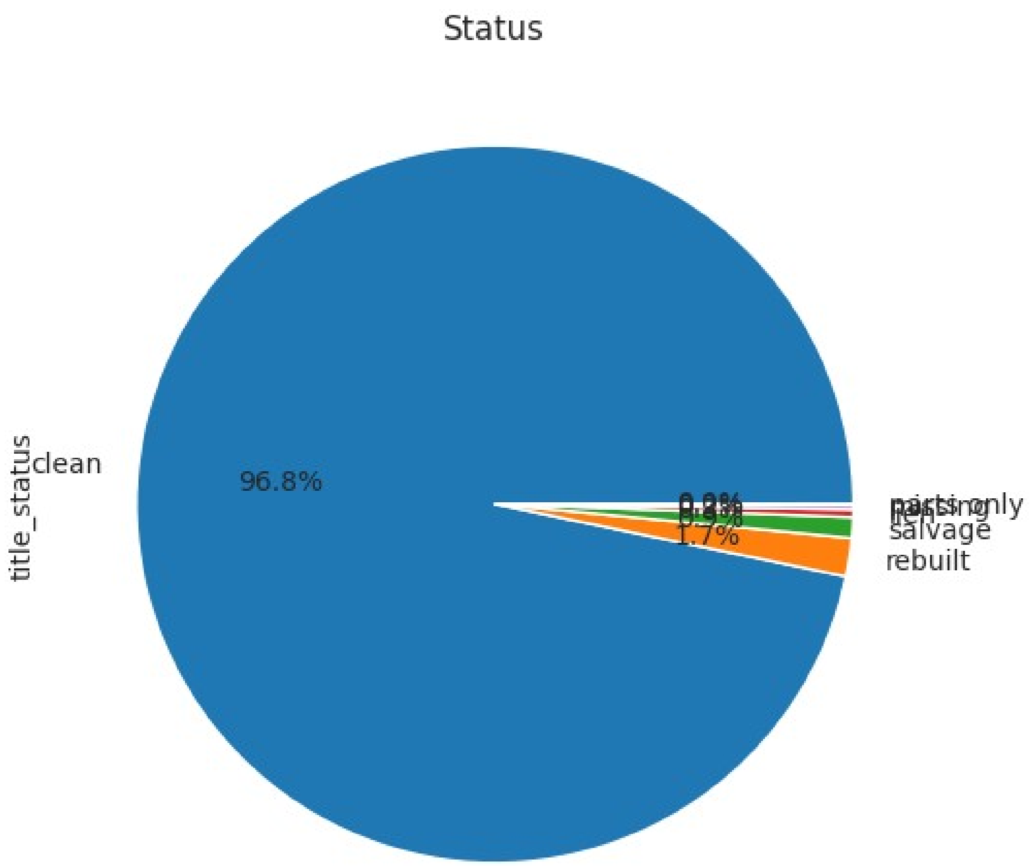
The graph shows the trend of hybrid and electric cars is slowly developing. Gas is the primary source to power for these vehicles still.

Q9. What is the status of the vechiles for sale?

```
df.title_status.unique()
```

```
array([nan, 'clean', 'rebuilt', 'lien', 'salvage', 'missing',
       'parts only'], dtype=object)
```

```
plt.figure(figsize=(8,6))
df['title_status'].value_counts().plot(kind = 'pie', autopct='%1.1f%
%',title='Status');
```



According to dataset that the owner has mentioned most of the vechiles are maintained clean.

```
jovian.commit
```

```
<function jovian.utils.commit.commit(message=None, files=[],
outputs=[], environment=None, privacy='auto', filename=None,
project=None, new_project=None, git_commit=False, git_message='auto',
require_write_access=False, **kwargs)>
```

Summary

1. The number of vehicles in the used vehicles market are pretty good in condition.
2. 79.3% of vehicles for sale are automatic vehicles.
3. Most of the vehicles uses gas as a fuel.
4. Ford company has highest number of used vechiles for sale and toyato vechiles are more expensive than others.
5. 96.8% of vechiles are cleanly maintained and 17% are rebuilt vechiles.

Future Work

1.Code optimization

2.Improving the documentation part of the project.

3.Adding more on visualization.

Reference

1.Dataset:Used Cars Dataset(Kaggale)

2.Opendatsets library: https://github.com/JovianML/opendatasets.

3.EDA project from scratch: https://www.youtube.com/watch?v=kLDTbavcmd0

```
jovian.commit
```

```
<function jovian.utils.commit.commit(message=None, files=[],
outputs=[], environment=None, privacy='auto', filename=None,
project=None, new_project=None, git_commit=False, git_message='auto',
require_write_access=False, **kwargs)>
```