MOHEB TODARY 900212311 ASSIGNMENT 1

This data set is all about giving observations on cars and giving it's price at the end. This could be used in machine learning; to predict the prices of the cars or to set a price to a new car according to its specifications. Throughout this paper, I will analyze cars data using a dataset named, "AutoData.csv" from kaggle.com. This dataset contains multiple observations of cars with its different specification including variables such as horsepower, fuelsystems and car dimentions. I will start by analyzing each continuous variable on its own then dig deeper into the correlations between relavent variables. Then I will explore the discrete variables and interpret them.

Intro to the data

In [191…

```
import pandas as pd
import numpy as np
df1 = pd.read_csv('AutoData.csv')
df2 = pd.read_csv('AutoData.csv')
df3 = pd.read_csv('AutoData.csv')
df4 = pd.read_csv('AutoData.csv')
df5 = pd.read_csv('AutoData.csv')

df = pd.concat([df1,df2,df3,df4,df5],ignore_index=True)


df.head()
```

Out[191…

	symboling	make	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	...	enginesize	fuelsystem	boreratio	stroke	compressionratio	horsepower	peakrpm
0	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	168.8	...	130	mpfi	3.47	2.68	9.0	111	5000
1	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	168.8	...	130	mpfi	3.47	2.68	9.0	111	5000
2	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	171.2	...	152	mpfi	2.68	3.47	9.0	154	5000
3	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	176.6	...	109	mpfi	3.19	3.40	10.0	102	5500
4	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	176.6	...	136	mpfi	3.19	3.40	8.0	115	5500

5 rows × 25 columns

In [2]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1025 entries, 0 to 204
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   symboling         1025 non-null   int64
 1   make              1025 non-null   object
 2   fueltype          1025 non-null   object
 3   aspiration        1025 non-null   object
 4   doornumber        1025 non-null   object
 5   carbody           1025 non-null   object
 6   drivewheel        1025 non-null   object
 7   enginelocation    1025 non-null   object
 8   wheelbase         1025 non-null   float64
 9   carlength         1025 non-null   float64
 10  carwidth          1025 non-null   float64
 11  carheight         1025 non-null   float64
 12  curbweight        1025 non-null   int64
 13  enginetype        1025 non-null   object
 14  cylindernumber    1025 non-null   object
 15  enginesize        1025 non-null   int64
 16  fuelsystem        1025 non-null   object
 17  boreratio         1025 non-null   float64
 18  stroke            1025 non-null   float64
 19  compressionratio  1025 non-null   float64
 20  horsepower        1025 non-null   int64
 21  peakrpm           1025 non-null   int64
 22  citympg           1025 non-null   int64
 23  highwaympg        1025 non-null   int64
 24  price             1025 non-null   float64
dtypes: float64(8), int64(7), object(10)
memory usage: 208.2+ KB
```

In [3]:

```
df.shape
```

Out[3]:

```
(1025, 25)
```

In [4]:

```
df.describe()
```

Out[4]:

	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	compressionratio	horsepower	peakrpm	citympg	highwaympg	pric
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.00000
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	3.329756	3.255415	10.142537	104.117073	5125.121951	25.219512	30.75122	13276.71057
std	1.242872	6.010003	12.313169	2.141010	2.438745	519.662255	41.561280	0.270314	0.312984	3.964275	39.466857	476.053119	6.529352	6.87298	7973.23383
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	2.540000	2.070000	7.000000	48.000000	4150.000000	13.000000	16.00000	5118.00000
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	3.150000	3.110000	8.600000	70.000000	4800.000000	19.000000	25.00000	7788.00000
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	3.310000	3.290000	9.000000	95.000000	5200.000000	24.000000	30.00000	10295.00000
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	3.580000	3.410000	9.400000	116.000000	5500.000000	30.000000	34.00000	16503.00000
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	3.940000	4.170000	23.000000	288.000000	6600.000000	49.000000	54.00000	45400.00000

Some distributions of continuous variables and their numerical summaries.

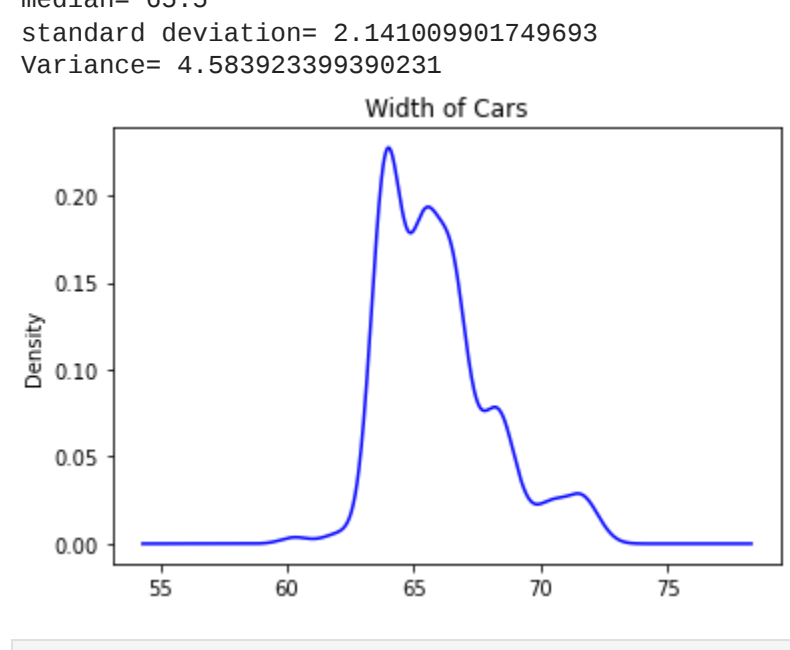
In [5]:

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

In [195…

```
df.carwidth.plot.density(color='blue', title='Width of Cars')
plt.show
print('mean=',df['carwidth'].mean())
print('median=',df['carwidth'].median())
print('standard deviation=',df['carwidth'].std())
print('Variance=',df['carwidth'].var())
```

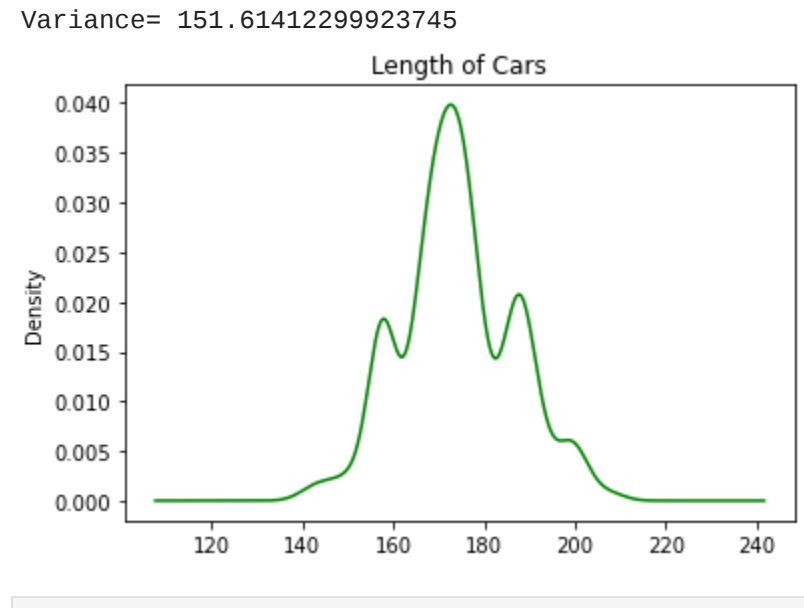
```
mean= 65.90780487804898
median= 65.5
standard deviation= 2.141009901749693
Variance= 4.583923399390231
```



In [221…

```
df.carlength.plot.density(color='green', title='Length of Cars')
plt.show
print('mean=',df['carlength'].mean())
print('median=',df['carlength'].median())
print('standard deviation=',df['carlength'].std())
print('Variance=',df['carlength'].var())
```

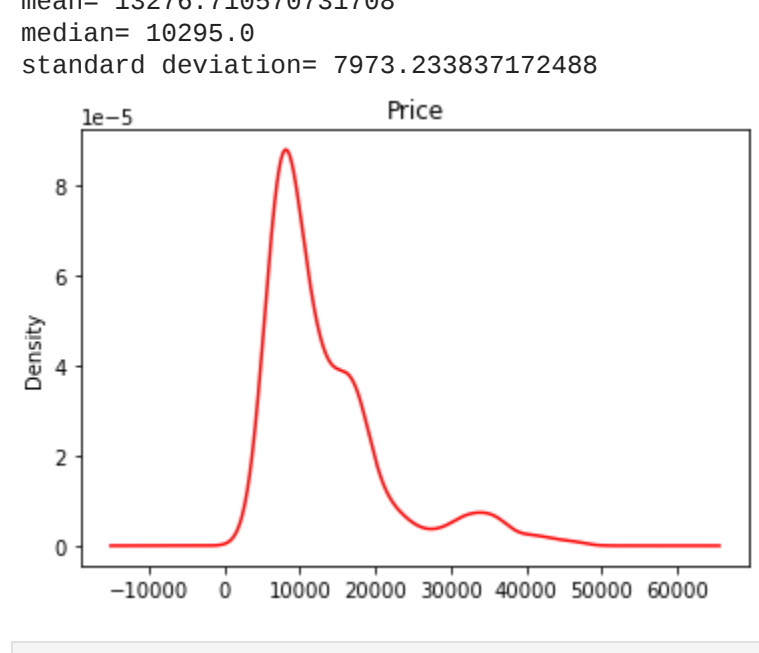
```
mean= 174.04926829268317
median= 173.2
standard deviation= 12.313168682318839
Variance= 151.61412299923745
```



In [192…

```
df.price.plot.density(color='red', title='Price')
plt.show
print('mean=',df['price'].mean())
print('median=',df['price'].median())
print('standard deviation=',df['price'].std())
```

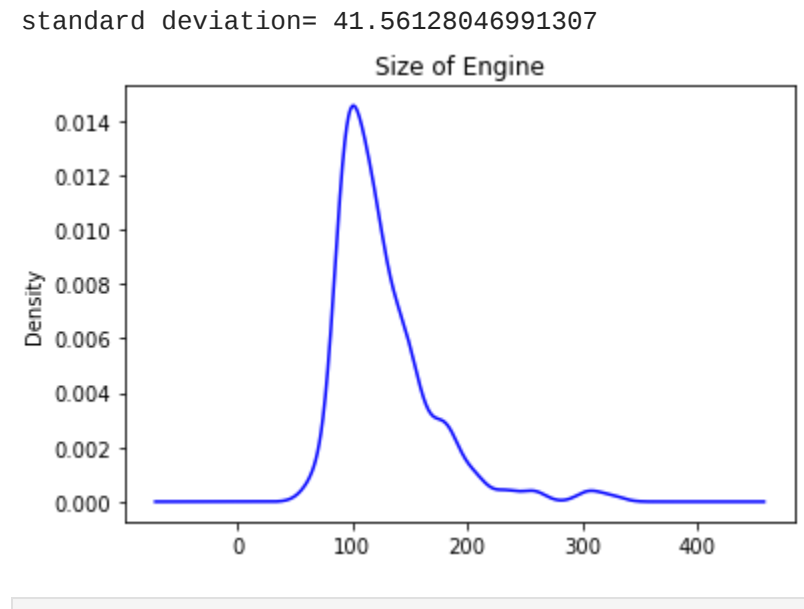
```
mean= 13276.710570731708
median= 10295.0
standard deviation= 7973.233837172488
```



In [196…

```
df.enginesize.plot.density(color='blue', title='Size of Engine')
plt.show
print('mean=',df['enginesize'].mean())
print('median=',df['enginesize'].median())
print('standard deviation=',df['enginesize'].std())
```

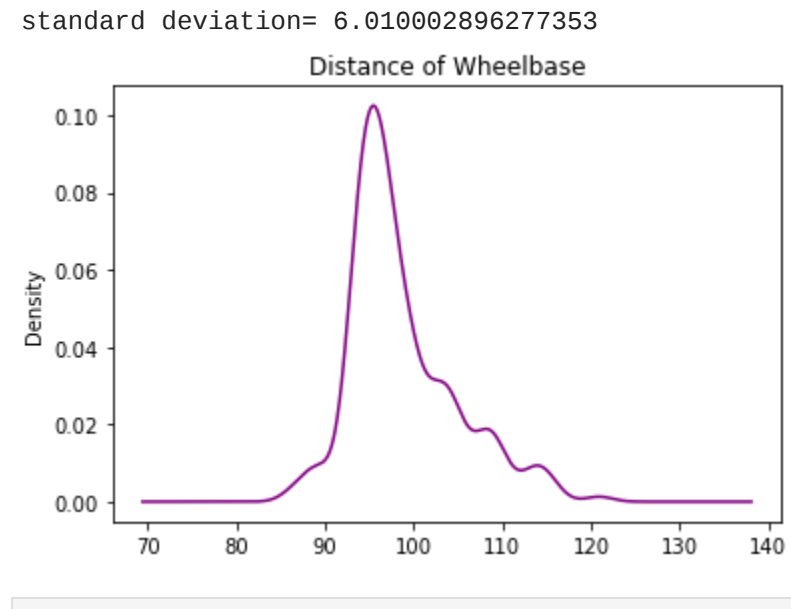
```
mean= 126.90731707317073
median= 120.0
standard deviation= 41.56128046991307
```



In [197…

```
df.wheelbase.plot.density(color='purple', title='Distance of Wheelbase')
plt.show
print('mean=',df['wheelbase'].mean())
print('median=',df['wheelbase'].median())
print('standard deviation=',df['wheelbase'].std())
```

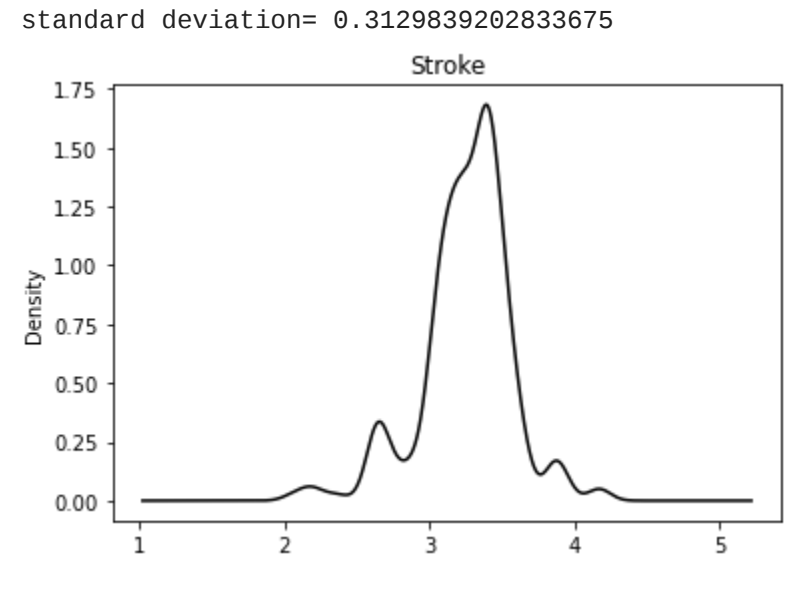
```
mean= 98.75658536585348
median= 97.0
standard deviation= 6.010002896277353
```



In [198…

```
df.stroke.plot.density(color='black', title='Stroke')
plt.show
print('mean=',df['stroke'].mean())
print('median=',df['stroke'].median())
print('standard deviation=',df['stroke'].std())
```

```
mean= 3.255414634146352
median= 3.29
standard deviation= 0.3129839202833675
```



Correlations between continuous variables

In [199…

```
b = df[["wheelbase", 'carlength','carheight','carwidth']].corr(method='pearson')
b
```

Out[199…

	wheelbase	carlength	carheight	carwidth
wheelbase	1.000000	0.874587	0.589435	0.795144
carlength	0.874587	1.000000	0.491029	0.841118
carheight	0.589435	0.491029	1.000000	0.279210
carwidth	0.795144	0.841118	0.279210	1.000000

In [44]:

```
sns.heatmap(b, annot=True)
```

Out[44]: <AxesSubplot:>

car length vs car width

In [6]:

```
x = df[["carlength", 'carwidth']].corr(method='pearson')
x
```

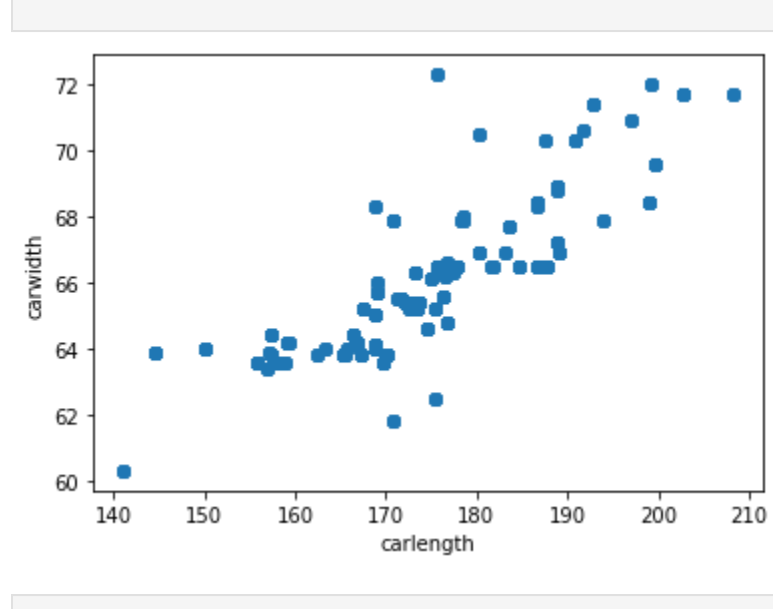
Out[6]:

	carlength	carwidth
carlength	1.000000	0.841118
carwidth	0.841118	1.000000

0.84 correlation means there is a strong positive corrleation between the car length and car width, this is because the number is closer to 1 than to zero. We could prove this by doing a scatter plot and observing the positive correlation with our eyes.

In [7]:

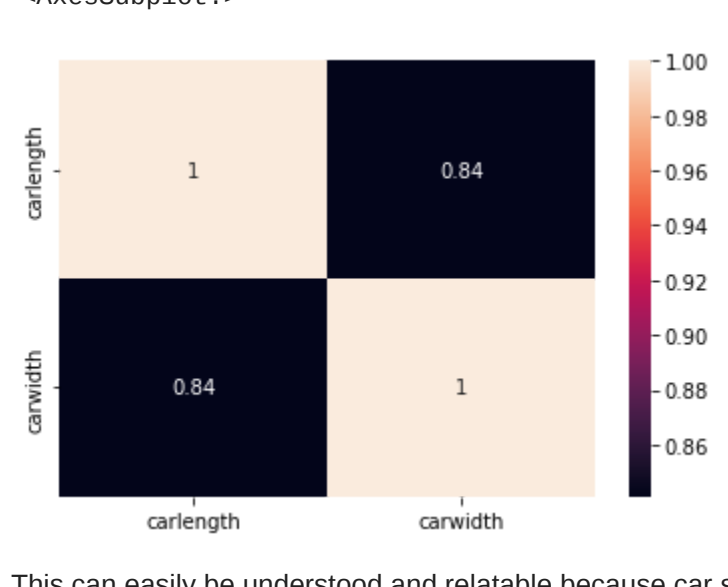
```
plt.scatter(df['carlength'],df['carwidth'])
plt.ylabel("carwidth")
plt.xlabel("carlength")
plt.show()
```



In [8]:

```
import seaborn as sns
sns.heatmap(x, annot=True)
```

Out[8]: <AxesSubplot:>



This can easily be understood and relatable because car sizes have a sort of propotion with its length and width not to be seen faulty. Hence, as the length of a car increses its width increases as well.

Highway Average Speed vs The cars weight

In [10]:

```
z = df[['highwaympg', 'curbweight']].corr(method='pearson').round(3)
z
```

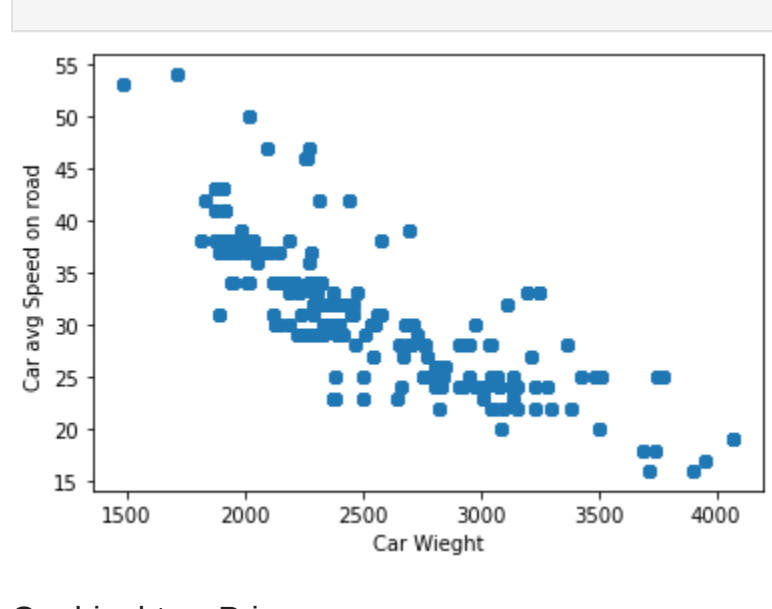
Out[10]:

	highwaympg	curbweight
highwaympg	1.000	-0.797
curbweight	-0.797	1.000

As seen in the table above, the correlation between the car's weight and the speed the car will reach while driving on an open road without stopping or starting, typically at a higher speed, is -0.797. This means that they are negatively linearly correlated, such that as the car weight increases its speed on road will decrease. I will again show the graph to easily observe the correlation by eye.

In [206…

```
plt.scatter(df['curbweight'], df['highwaympg'])
plt.xlabel("Car Wieght")
plt.ylabel("Car avg Speed on road ")
plt.show()
```



Car hieght vs Price

In [25]:

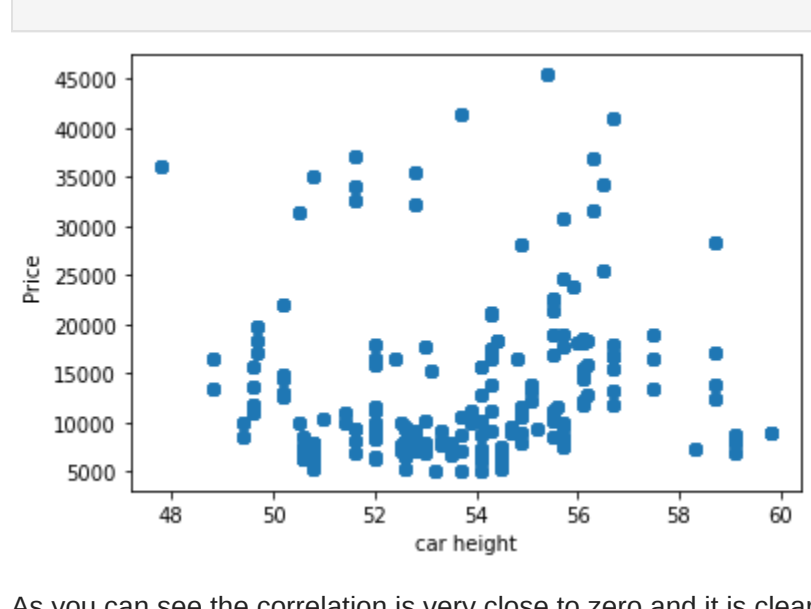
```
c = df[['price', 'carheight']].corr(method='pearson').round(3)
c
```

Out[25]:

	price	carheight
price	1.000	0.119
carheight	0.119	1.000

In [27]:

```
plt.scatter(df['carheight'], df['price'])
plt.xlabel("car height")
plt.ylabel("Price ")
plt.show()
```



As you can see the correlation is very close to zero and it is clear on the graph that there is no clear leniar relationship. In fact, if we have the car height only we could not predict it's price as there is almost Zero correlation. This may be because height does not determine the quality of the car at all as quality determines the price in most cases.

Descrete variables

In [92]:

```
df
```

Out[92]:

	symboling	make	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	...	enginesize	fuelsystem	boreratio	stroke	compressionratio	horsepower	peakrpm
0	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	168.8	...	130	mpfi	3.47	2.68	9.0	111	5000
1	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	168.8	...	130	mpfi	3.47	2.68	9.0	111	5000
2	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	171.2	...	152	mpfi	2.68	3.47	9.0	154	5000
3	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	176.6	...	109	mpfi	3.19	3.40	10.0	102	5500
4	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	176.6	...	136	mpfi	3.19	3.40	8.0	115	5500
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	front	109.1	188.8	...	141	mpfi	3.78	3.15	9.5	114	5400
201	-1	volvo 144ea	gas	turbo	four	sedan	rwd	front	109.1	188.8	...	141	mpfi	3.78	3.15	8.7	160	5300
202	-1	volvo 244dl	gas	std	four	sedan	rwd	front	109.1	188.8	...	173	mpfi	3.58	2.87	8.8	134	5500
203	-1	volvo 246	diesel	turbo	four	sedan	rwd	front	109.1	188.8	...	145	idi	3.01	3.40	23.0	106	4800
204	-1	volvo 264gl	gas	turbo	four	sedan	rwd	front	109.1	188.8	...	141	mpfi	3.78	3.15	9.5	114	5400

1025 rows × 25 columns

drivewheel= four-wheel drive, front-wheel drive, and rear-wheel drive.

In [94]:

```
df['drivewheel'].value_counts()
```

Out[94]:

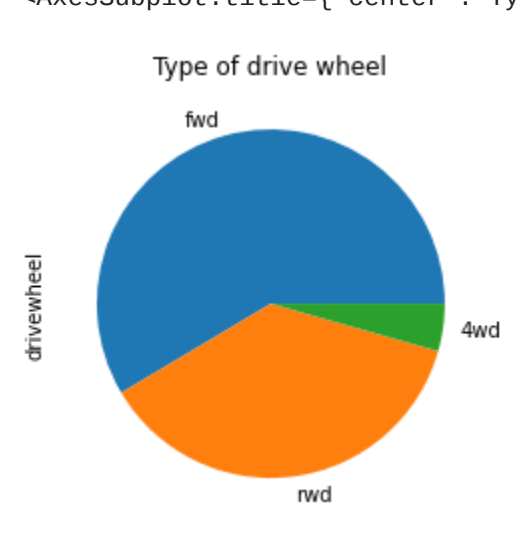
```
fwd    600
rwd    380
4wd     45
Name: drivewheel, dtype: int64
```

This means we can categorise the variable into three groups which are front wheel, rear-wheel, and four wheel. Instead of just getting numbers we could use graphs such as pie charts and barcharts to easily see the differences and interpret them.

In [96]:

```
df['drivewheel'].value_counts().plot(kind='pie', title='Type of drive wheel')
```

Out[96]: <AxesSubplot:title={'center':'Type of drive wheel'}, ylabel='drivewheel'>



Front Wheel takes up the majority of the piechart which means that most cars have a foward drive wheel system.

Car body type= sedan, hatchback, wagon, hardtop, and convertable

In [184…

```
df['carbody'].value_counts(normalize=True)
```

Out[184…

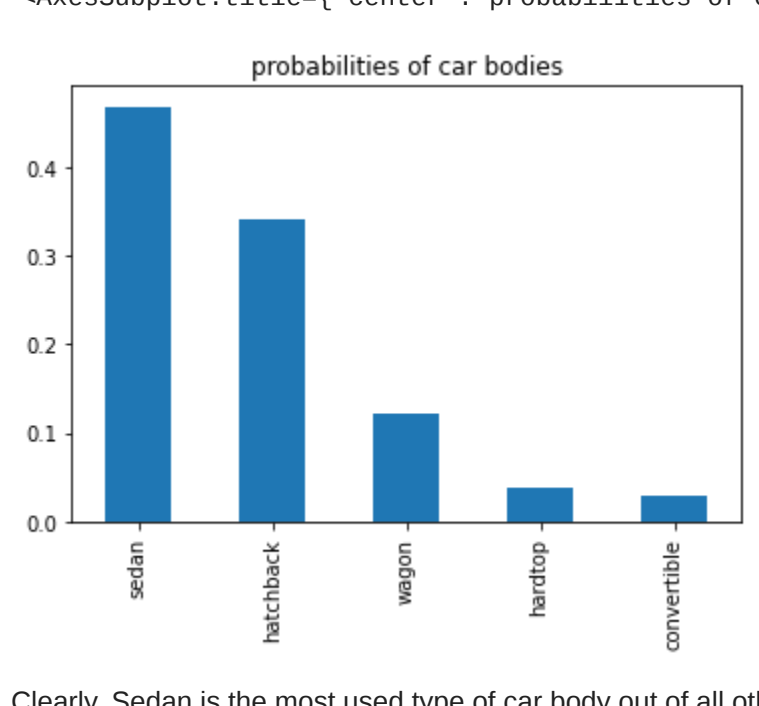
```
sedan          0.468293
hatchback      0.341463
wagon          0.121951
hardtop        0.039024
convertible    0.029268
Name: carbody, dtype: float64
```

This shows the percentages of the occurance of each type of car body.

In [190…

```
df['carbody'].value_counts(normalize=True).plot(kind='bar', title= 'probabilities of car bodies')
```

Out[190… <AxesSubplot:title={'center':'probabilities of car bodies'}>



Clearly, Sedan is the most used type of car body out of all other types, followed by hatchback. While, on the other hand the convertable types are the rareset.

Number of doors vs The body of the car

In [117…

```
o= pd.crosstab(df['carbody'], df['doornumber'])
o
```

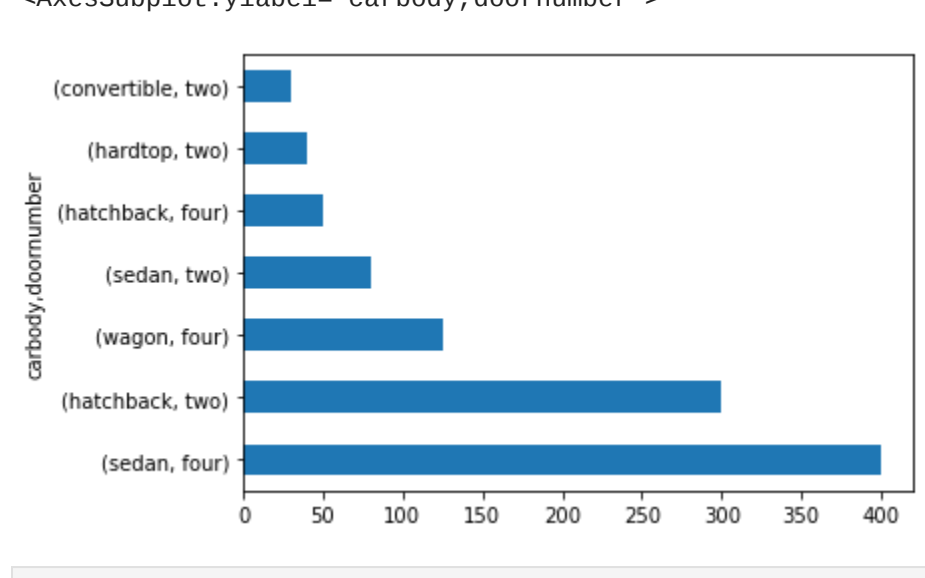
Out[117…

doornumber	four	two
carbody		
convertible	0	30
hardtop	0	40
hatchback	50	300
sedan	400	80
wagon	125	0

In [112…

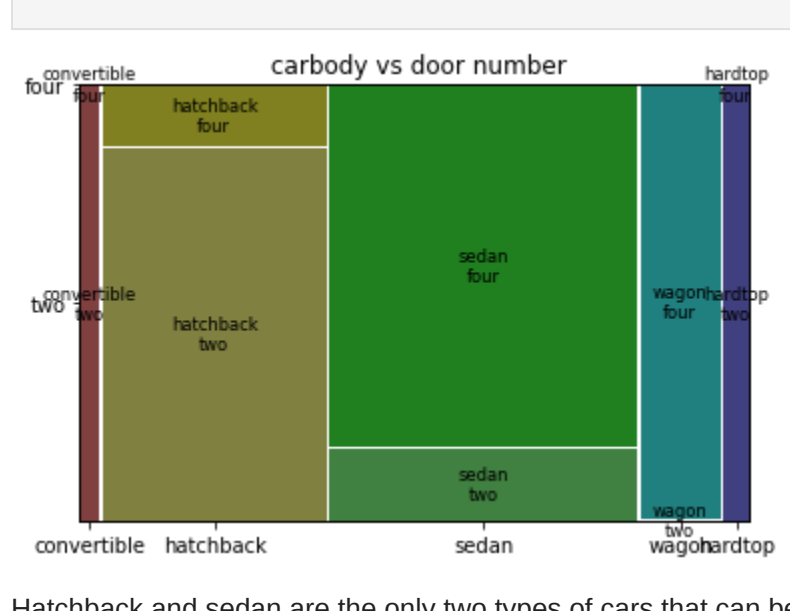
```
df[['carbody','doornumber']].value_counts().plot(kind='barh')
```

Out[112… <AxesSubplot:ylabel='carbody,doornumber'>



In [222…

```
from statsmodels.graphics.mosaicplot import mosaic
mosaic(df, ['carbody', 'doornumber'], title= ' carbody vs door number')
plt.show()
```



Hatchback and sedan are the only two types of cars that can be made with both 2 or 4 doors. However, hatchbacks are more likely designed with 2 doors; while sedan is far less likely designed with 2 doors.

Fuel type vs Engine location

In [210…

```
pd.crosstab(df['fueltype'], df['enginelocation'])
```

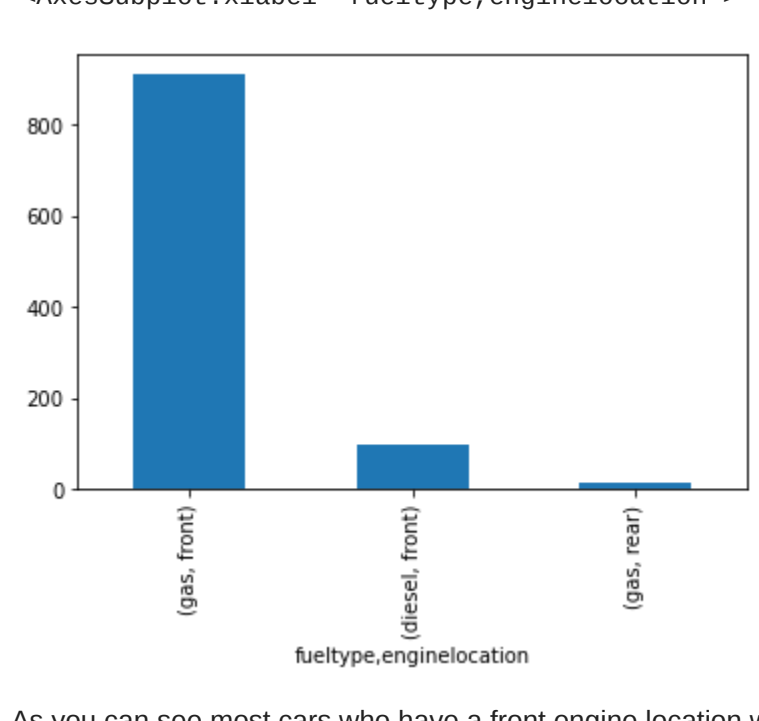
Out[210…

enginelocation	front	rear
fueltype		
diesel	100	0
gas	910	15

In [212…

```
df[['fueltype','enginelocation']].value_counts().plot(kind='bar')
```

Out[212… <AxesSubplot:xlabel='fueltype,enginelocation'>



As you can see most cars who have a front engine location work with gas and not deisel. In addition, there is not a single car where the engine is placed at the back and uses fuel type diesel.

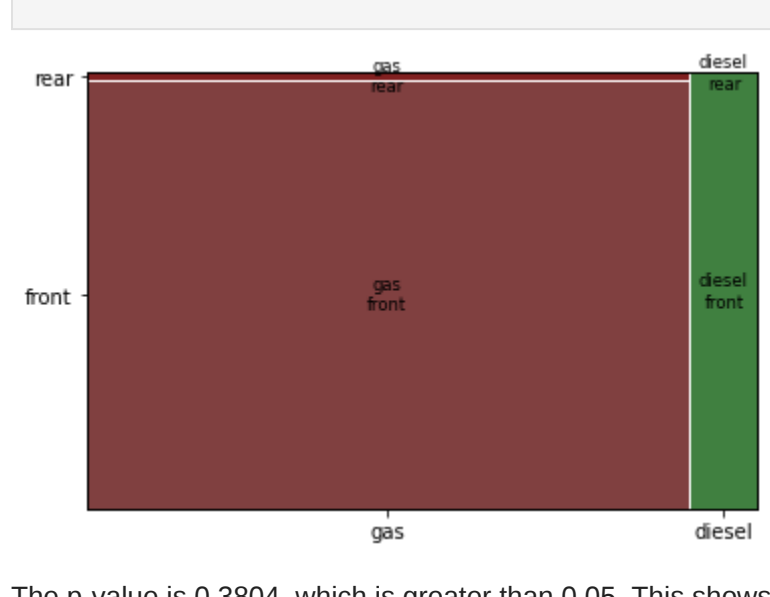
In [150…

```
o= pd.crosstab(df['fueltype'], df['enginelocation'])
o
import statsmodels.api as sm
Table = sm.stats.Table(o)
print(Table)
Table.table
Table.fittedvalues
Table.resid_pearson
rslt= Table.test_nominal_association()
print(rslt.pvalue)
```

```
A 2x2 contingency table with counts:
[[1.0e+02 5.0e-01]
 [9.1e+02 1.5e+01]]
0.38037245860225666
```

In [151…

```
from statsmodels.graphics.mosaicplot import mosaic
mosaic(df, ['fueltype', 'enginelocation'])
plt.show()
```



The p-value is 0.3804, which is greater than 0.05. This shows that there is no correlation between fuel type and engine location, whether it is the front or the back of a car. In other words, we agree with the null hypothesis that there arn't related.

Fuel type vs engine location

In [218…

```
o= pd.crosstab(df['fuelsystem'], df['enginelocation'])
o
```

Out[218…

enginelocation	front	rear
fuelsystem		
1bbl	55	0
2bbl	330	0
4bbl	15	0
idi	100	0
mfi	5	0
mpfi	455	15
spdi	45	0
spfi	5	0

Note that the engine is not located in the frontg only when the fuel system is mpfi

In [214…

```
o= pd.crosstab(df['fuelsystem'], df['enginelocation'])
o
import statsmodels.api as sm
Table = sm.stats.Table(o)
print(Table)
Table.table
Table.fittedvalues
Table.resid_pearson
rslt= Table.test_nominal_association()
print(rslt.pvalue)
```

```
A 8x2 contingency table with counts:
[[ 55.    0.5]
 [330.    0.5]
 [ 15.    0.5]
 [100.    0.5]
 [  5.    0.5]
 [455.   15. ]
 [ 45.    0.5]
 [  5.    0.5]]
0.03508084709462367
```

The p-value is less than 0.05; therefore, we reject the null hypothesis and we can conclude that the fuel system and enginelocations are somehow correlated.

Thank you ❤️