# پروژه صفر

## پرکردن نواقص یک دیتاست با پیشبینی آن ها به کمک تحلیل آماری

سینا سلیمیان

810197528

توضیح کلی پروژه:

در این پروژه dataset ای مربو به 205 خودرو به ما داده شده است که شامل انواع مختلف خودرو و اطلاعاتی منحصر به فرد برای هر کدام می باشد. برخی از این اطلاعات برای برخی از خودرو ها نا موجود می باشد از جمله قیمت خودرو. هدف ما تخمین قیمت احتمالی این خودرو ها می باشد که این کار را با سایر اطلاعات موجود از آن خودرو می خواهیم انجام دهیم.

**بخش اول**: داده ی ورودی ما یک فایل با پسوند csv است که در ابتدای کار آن را خوانده و به کمک کتابخانه pandas در یک Data Frame ذخیره می کنیم.

Head: 5 داده ی اول در آرایه

Tail: 5 داده ی آخر در آرایه

Describe: count (تعداد داده های غیر NA/null)

max (ماکسیمم داده در آن ستون)

min (مینیمم داده در آن ستون)

std (انحراف معیار)

25%: چارک اول

50%: چارک دوم(میانه)

75%: چارک سوم

**بخش دوم:** با استفاده از تابع info نوع هر داده(ستون ها) مشخص می شود. برای داده ی fueltype، به کمک تابع replace، مقدار gas را با 0 و diesel را با 1 جایگذاری می کنیم و همچنین برای ستون cilyndernumber، مقادیر را به صورت عددی تغییر می دهیم.

برای ستون CarName، چون تنوع خودرو ها زیاد است، به جای استفاده از replace، از Label Encoding استفاده می کنیم. در این تکنیک، به هر نوع جدید از خودرو یک id جدید نسبت داده می شود و به این صورت به جای کار با نام خودرو ها، با id آن ها کار می کنیم.

**بخش سوم**: به کمک تابع isna() تعداد داده های Nan هر ستون را بدست می آوریم و به کمک تابع fillna() این مقادیر را با mean(). پر می کنیم.

مزایا: باعث می شود میانگین کلی داده ها تغییری نکند و بتوان با احتمال خوبی از آن داده در صورت نیاز استفاده کرد و همچنین اگر بتوانیم

مقدار Nan را پر کنیم، از سایر ویژگی های دقیق خودرو نیز می توانیم استفاده کنیم و بی استفاده نمی شوند.

معایب: به دلیل دقیق نبودن داده ی مورد نظر، ممکن است از آن داده به عنوان داده دقیق استفاده کرد و این استفاده برایمان مشکل ساز شود. می دانیم اگر میانگین چند عدد را داشته باشیم بی نهایت حالت برای اعداد وجود خواهد داشت و ممکن است تفاوت میان آن ها خیلی کن و خیلی زیاد باشد.

به کمک تابع dropna() سطر هایی که قیمت آن ها Nan می باشد را جدا می کنیم و در DataFrame عه Nan_prices ذخیره می کنیم.

**بخش چهارم:** به کمک تابع value_counts() تعداد خودرو برای هر cilyndernumber را پیدا می کنیم.

**بخش پنجم:** با فیلتر کردن داده ها به کمک عملیات های Boolean ای به تعداد 3 خودرو با Car_id 50و74و75 می رسیم.

| | car_ID | Car Name | fuel type | carlength | car width | carh eight | curb weight | cylinder number | engin esize | horse power | city mpg | highw aympg | price |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **49** | 50 | 50 | 0 | 191.7 | 70.6 | 47.8 | 3950 | 12 | 326.000000 | 262.000000 | 13 | 17 | 36000.0 |
| **73** | 74 | 18 | 0 | 208.1 | 71.7 | 56.7 | 3900 | 8 | 125.418848 | 105.747253 | 14 | 16 | 40960.0 |
| **74** | 75 | 21 | 0 | 199.2 | 72.0 | 55.4 | 3715 | 8 | 304.000000 | 184.000000 | 14 | 16 | 45400.0 |

**بخش ششم**: داده ها را با تابع ()eq فیلتر می کنیم و سپس ()mean را روی price آن ها صدا می زنیم.

```
gas mean:     13170.738957831325
diesel mean:  15326.894736842105
time is:      0.01123046875
```

**بخش هفتم:** به کمک حلقه ی for قیمت خودرو ها را با یکدیگر جمع کرده و تقسیم بر تعدادشان می کنیم تا میانگین بدست آید.

```
gas mean:     13170.738957831325
diesel mean:  15326.894736842105
time is:      0.034984588623046875
```

با تقسیم زمان اجرای دو بخش بر یکدیگر متوجه می شویم که در حالتی که از vrctorization استفاده می کنیم، سرعت 3.1 برابر حالتی است که از حلقه for استفاده می کنیم.

```
Speed difference: 3.1x
```

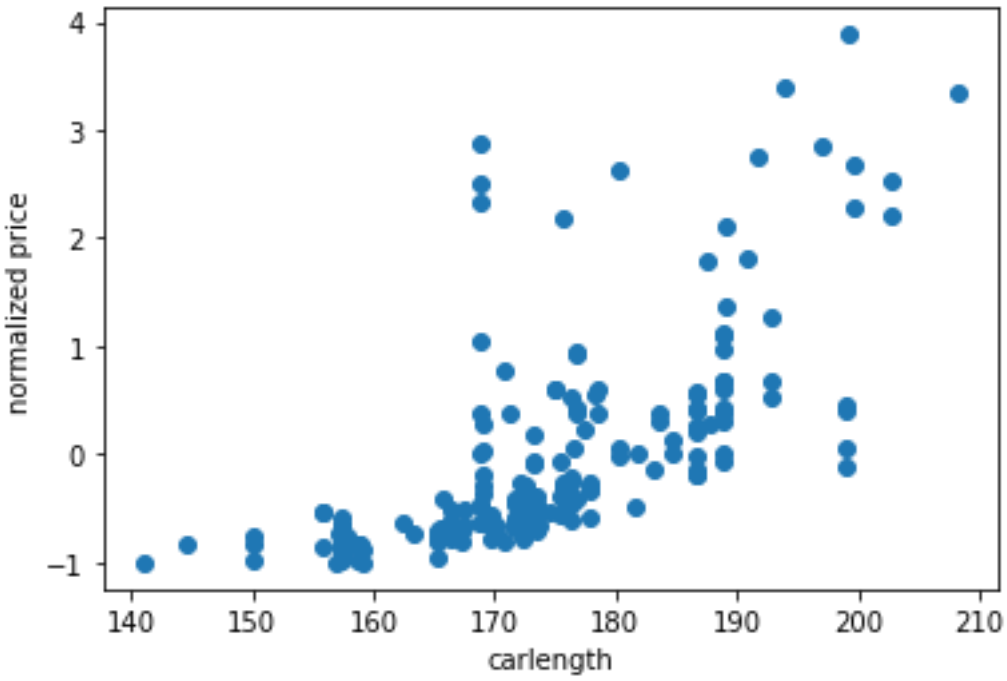
**بخش هشتم:** به کمک تابع ()hist نمودار توزیع هر ویژگی را نشان می دهیم. به کمک این نمودار می توان میزان پراکندگی داده ها را در یک ویژگی مشخص مشاهده کرد و یک تجسمی از فراوانی داده ها و میانگین آن ها پیدا کرد.

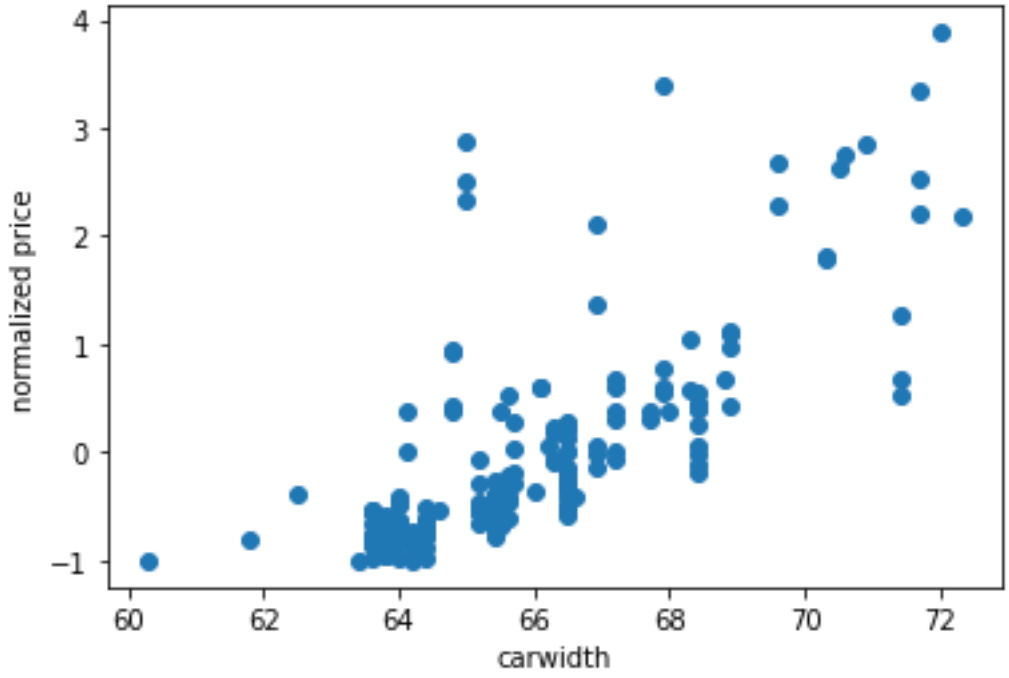
در آخر ستون های fueltype و CarName را که مقدار عددی ندارند، از data حذف می کنیم. و در NewData ذخیره می کنیم و از این به بعد با NewData کار می کنیم.

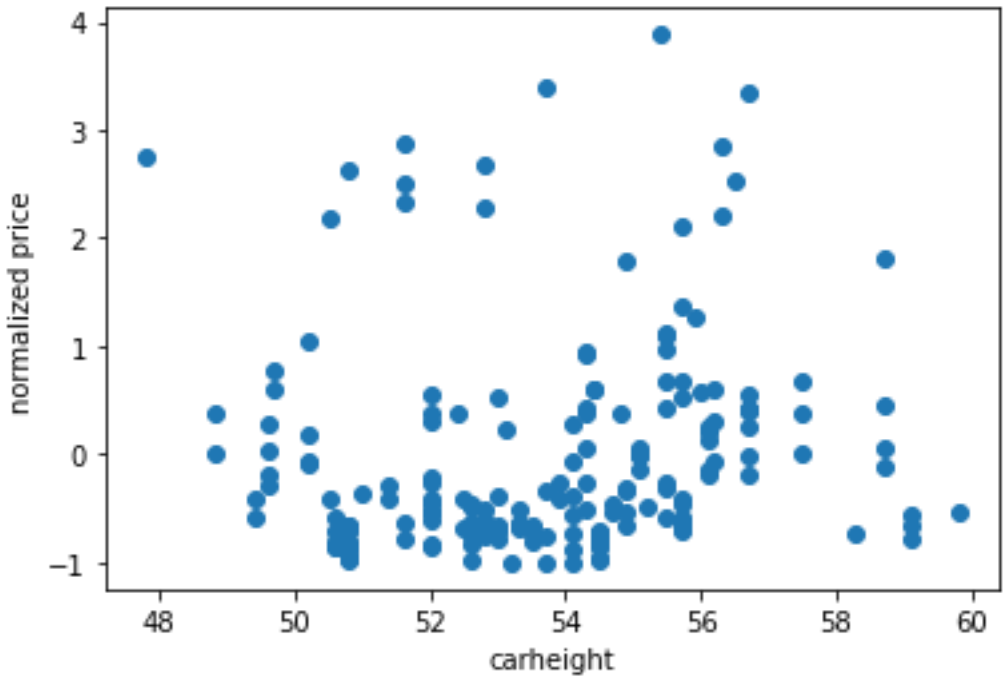
**بخش نهم**: میانگین و انحراف معیار را با ()mean و ()std پیدا کرده و قیمت ها را نرمال سازی می کنیم.

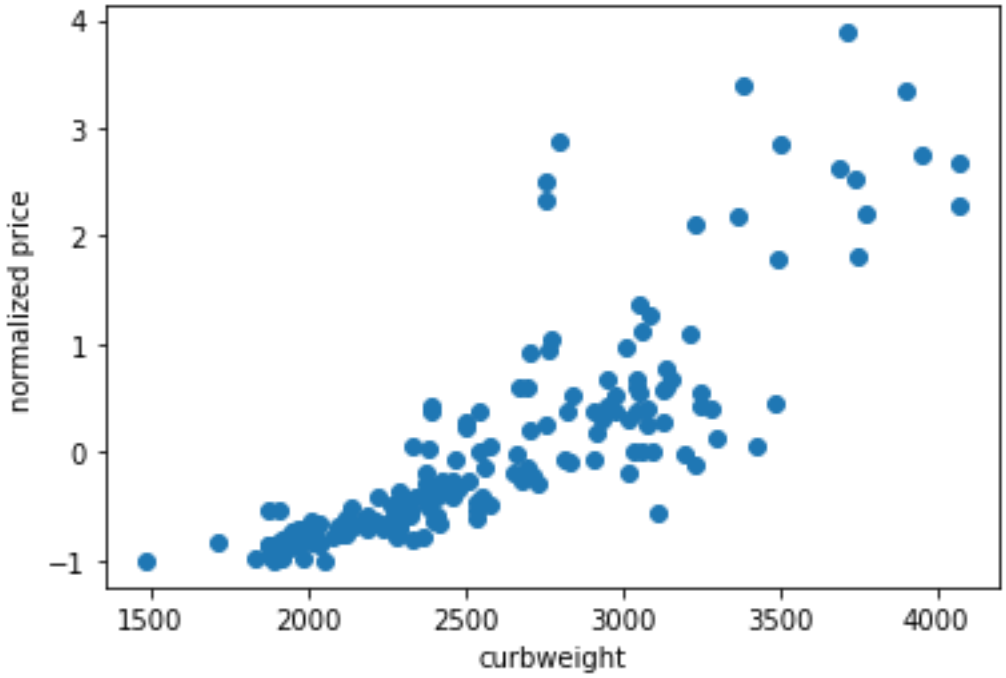
```
0      0.012484
1      0.377353
2      0.377353
3      0.067731
5      0.225577
         ...
200    0.419243
201    0.686368
202    0.982634
203    1.102234
204    1.121054
```

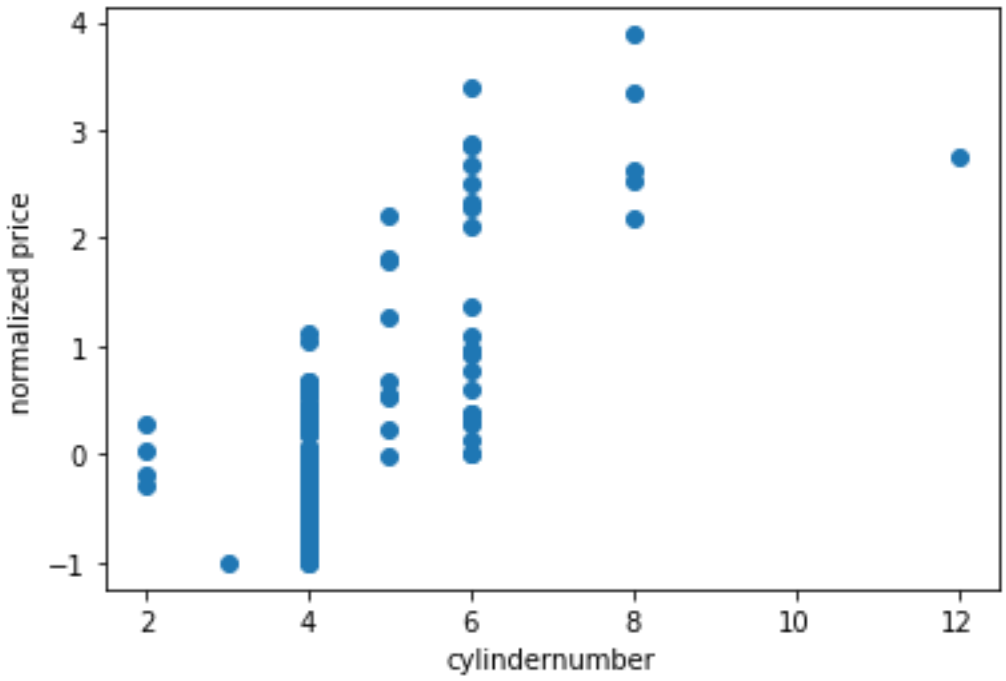
**بخش دهم**: برای پیدا کردن ویژگی با بیشترین همبستگی، نمودار های هر ویژگی را رسم می کنیم تا بتوانیم بهتر مقایسه کنیم.

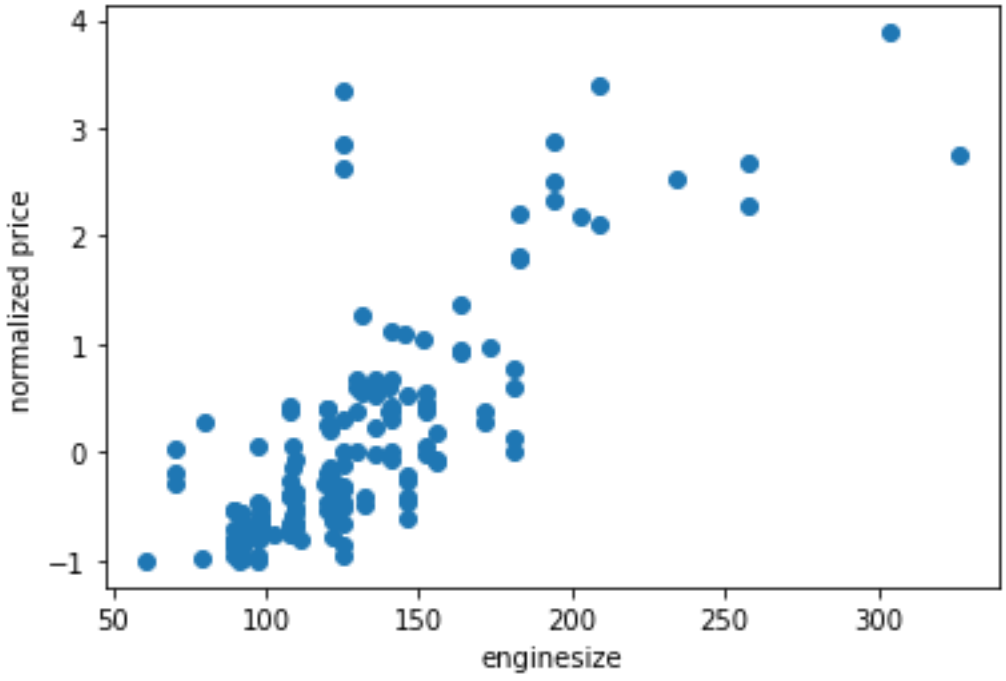

normalized price
carlength
4
3
2
1
0
−1
140
150
160
170
180
190
200
210

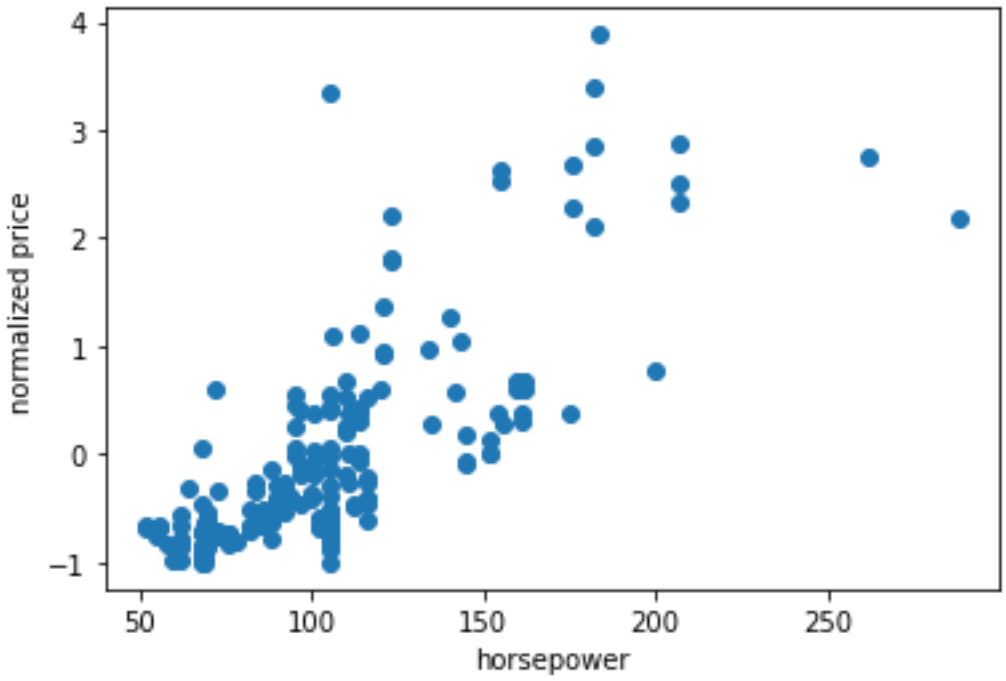

normalized price
carwidth
4
3
2
1
0
−1
60
62
64
66
68
70
72

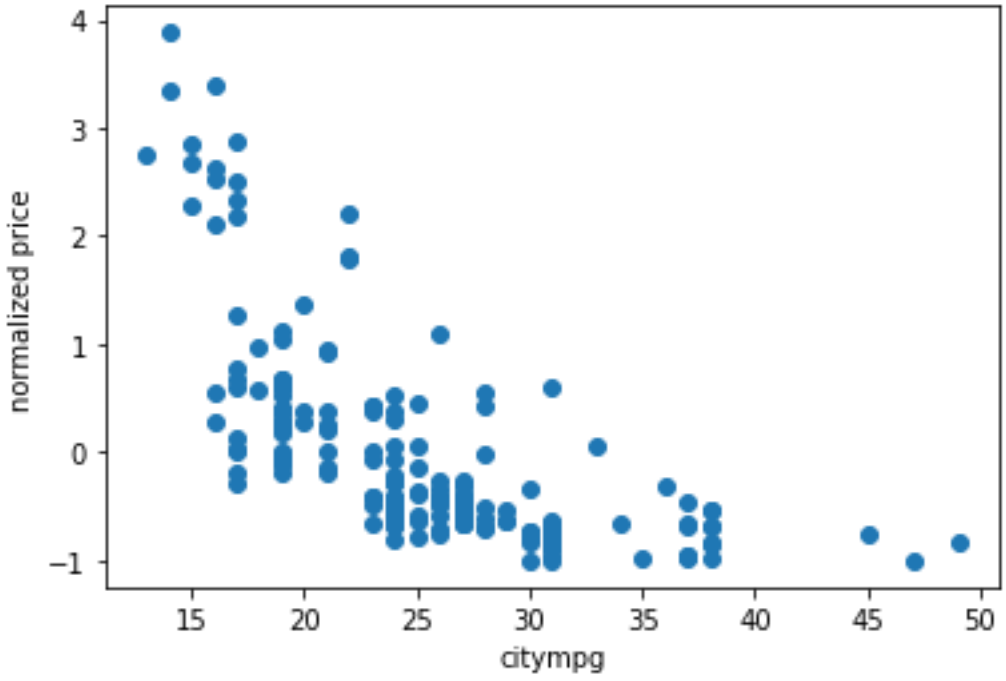

normalized price
carheight
4
3
2
1
0
−1
48
50
52
54
56
58
60

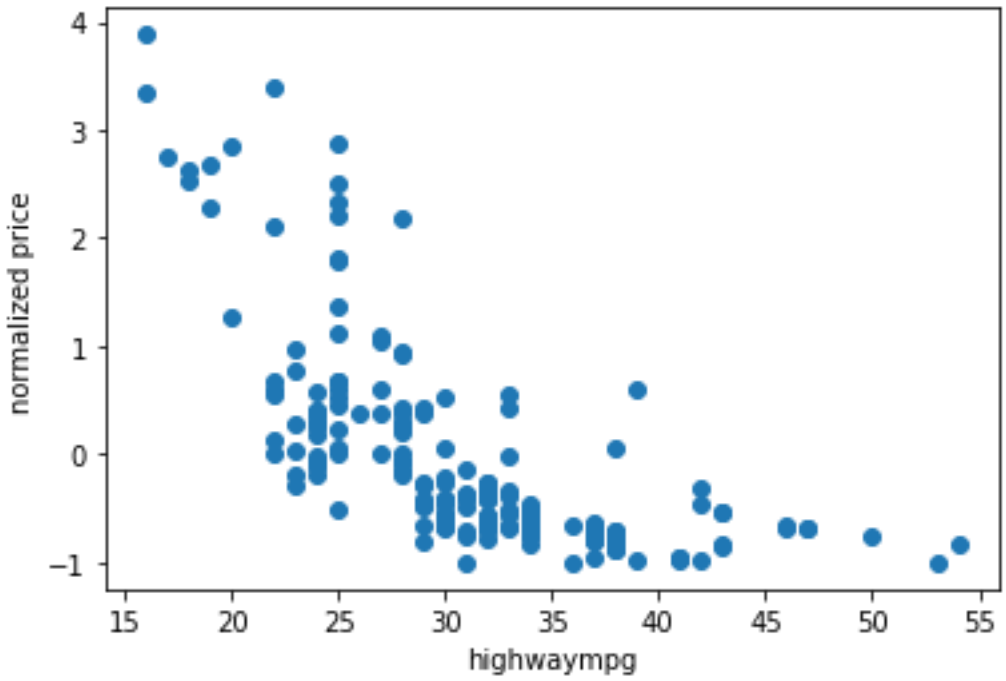

normalized price
curbweight
4
3
2
1
0
−1
1500
2000
2500
3000
3500
4000


normalized price
cylindernumber
4
3
2
1
0
−1
2
4
6
8
10
12


normalized price
enginesize
4
3
2
1
0
−1
50
100
150
200
250
300


normalized price
horsepower
4
3
2
1
0
−1
50
100
150
200
250


normalized price
citympg
4
3
2
1
0
−1
15
20
25
30
35
40
45
50




با مقایسه ی ویژگی ها، مشاهده می کنیم که ویژگی enginesize به نسبت سایرین همبستگی بیشتری با قیمت دارد و نمودار آن خطی تر می باشد و می توانیم خطی فرضی میان داده ها داشته باشیم که داده ها از آن خط خیلی فاصله نداند و پراکندگی شان کمتر است.

**بخش یازدهم**: یک DataFrame جدید با دو ستون enginesize و price می سازیم( EnginePrice_df )

محاسبه teta_1 و teta_0 :

برای محاسبه شیب نمودار، اگر دقت کنیم می بینیم که کمترین داده و بیشترین داده روی خط فرضی قرار می گیرند که بین تمام داده ها می باشد. پس شیب بین این رو نقطه را حساب می کنیم و برابر با teta_1 قرار می دهیم.

teta_1 = (max price – min price)/(max enginesize – min enginesize)

```
teta_1 = 0.018456839711072422
```

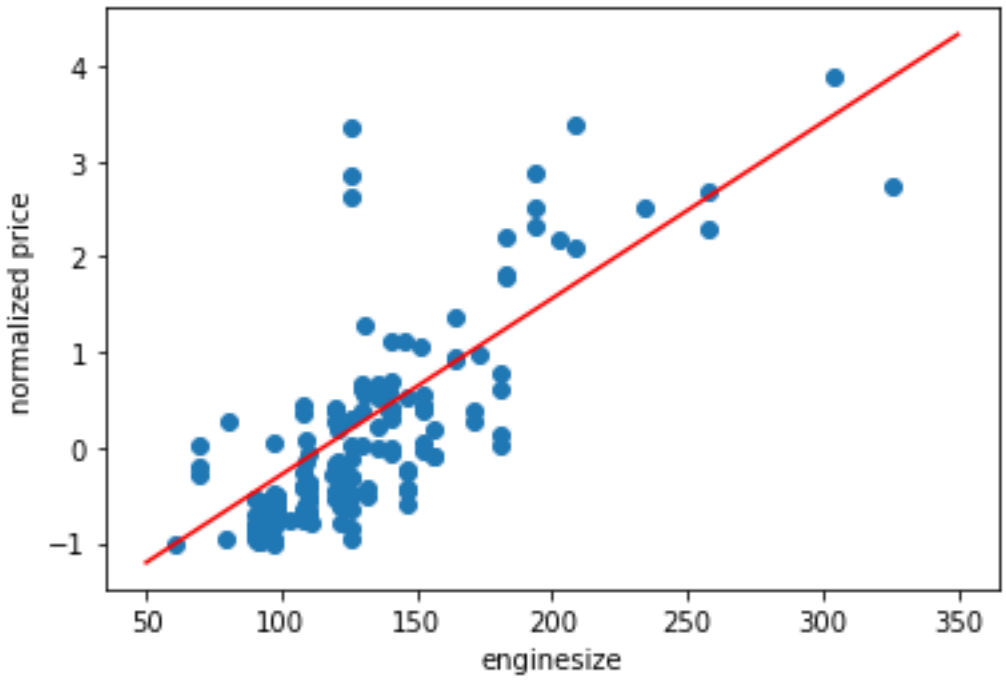
برای محاسبه ی teta_0 نقطه ی کوچکترین داده را داخل فرمول خط قرار می دهیم.

teta_0 = -teta_1 * (min enginesize) + (min price)

```
teta_0 = -2.130522934925918
```

**بخش دوازدهم**: تابع MSE را می سازیم تا میزان خطای تخمین تابع H_teta را محاسبه کنیم که می بینیم این مقدار برار 0.4 می باشد.

**بخش سیزدهم**: برای مقایسه تابع تخمین گر و مقادیر واقعی قیمت بر حسب enginesize، آن ها را در یک نمودار رسم می کنیم و می بینیم که تقریبا در یک راستا قرار دارند و نزدیک به یکدیگر هستند و تابع خوبی برای تخمین قیمت می باشد.



**بخش چهاردهم**: حال که از دقت تابع مورد نظرمان مطمین شدیم، می توانیم از آن در تخمین قیمت های Nan_prices استفاده کنیم.

```
car_ID    price
5      16518.566038
30     19558.716981
32      9830.233962
53      9678.226415
54      9678.226415
60     14390.460377
63     14390.460377
68     23662.920755
79      9830.233962
85     19558.716981
88     12566.369811
89     12566.369811
105    23358.905660
124    14390.460377
137    14910.151101
138    14238.452830
142    12262.354717
150    12262.354717
182    20318.754717
190    12414.362264
```

**نتیجه**: به کمک داده هایی که همبستگی خوبی با داده های مطلوب دارند، می توان با توجه به سوابق گذشته، داده مطلوب را تخمین زد.

**منابع**:


Geeksforgeeks.com

Stackoverflow.com

numpy.org

pandas.pydata.org

matplotlib.org