در فایل .ipynb ارسال شده ، توضیحات در هر مرحله کامنت شده اند ؛ این جا هم به طور خلاصه روند کد ها و سوالات

تئوری را بررسی می کنیم . (بعضی جاها تصاویری هم در این فایل pdf آورده ایم اما تصاویر جامع تر و مراحل کامل

در خود فایل .ipynb موجود است .)

تبدیل ویژگی های دسته ای به عددی :

بعضی از ویژگی ها که مشخص است ارتباطی با قسمت ندارند مثل car\_id و ... ؛ ما در این سوال به طور نمونه ویژگی

تعداد سیلندرها که ویژگی مهمی است را تبدیل به ویژگی عددی کرده ایم : (با مپ کردن ) توضیحات و تغییرات

به وجود آمده در data set هم پس از این تغییر نشان داده شده است .

در مراحل بعد ویژگی های دسته ای را drop کرده ایم .

قسمت اول رگرسیون خطی تک متغیره :

برای سوال 1 که قرار است فقط با 1 ویژگی مقدار price را حدس بزنیم ، ما ویژگی horsepower را انتخاب کرده ایم .

از horsepower و price کپی گرفته ایم و آن ها را تحت 2 آرایه با نام های horse , price به الگوریتم پیاده شده

داده ایم .پس از اجرای الگوریتم مقدار پارامتر های teta0 , teta1 و نمودار پیش بینی شده آورده شده است .

قسمت دوم رگرسیون خطی چند متغیره :

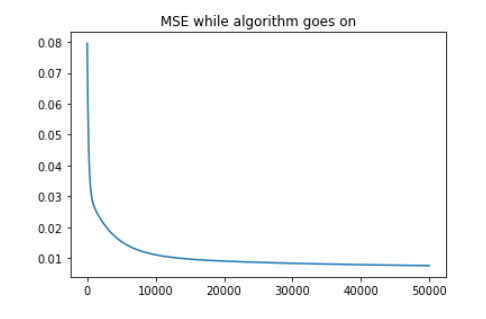
با توجه به تبدیل ویژگی دسته ای تعداد سیلندر به ویژگی عددی و داشتن ویژگی های عددی قبلی در مجوع 14 ویژگی

عددی داریم و نیاز به 15 پارامتر teta داریم ، این موضوع انجام شده است . پس از normalize کردن ویژگی ها و

قیمت نهایی ، از توابعی که پیاده سازی کرده ایم ، استفاده شده و مقدار این 15 پارامتر به همراه نمودار

original normalized price vs predicted value آورده شده است . می بینیم که پیش بینی موفق بوده است.

شکل زیر هم نمودار mse را برای تمام مراحل الگوریتم نشان می دهد .



دیگر نمودار ها هم در هر سناریو در فایل .ipynb موجود اند .

مقدار r2\_score هم عدد بالا و نزدیک به 1 می باشد (0.80) یعنی پیش بینی راضی کننده است . این موارد در قسمت

ارزیابی مدل رگرسیون خطی جزئی تر بررسی شده اند

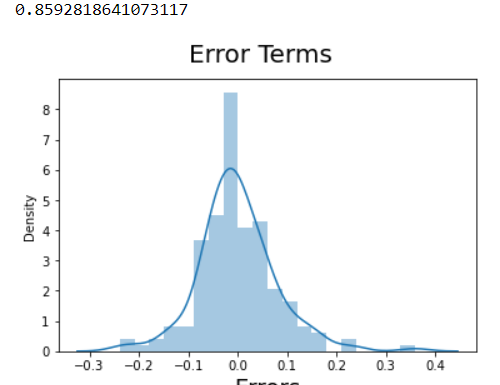
در مرحله ی بعد ما با توابعی که در کتابخانه ی پایتون آورده شده ، مثل import statsmodels.api هم پیش بینی

را انجام می دهیم تا صحت و شباهت عملکرد تابع پیاده سازی خود را بررسی کرده باشیم .

داده ها را به دوقسمت test and train قسمت می کنیم .

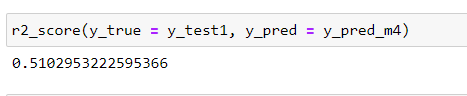
df\_train, df\_test = train\_test\_split(df, train\_size = 0.8, test\_size = 0.2, random\_state = 100)

و عملکرد توابع خود پایتون هم شبیه عملکرد توابع پیاده سازی شده است .



پس از این قسمت عملکرد مدل ساخته شده توسط خود پایتون و مقدار خروجی r2\_score را می بینیم که عدد نسبتا پایینی

است (0.5) دلیل این موارد را در قسمت ارزیابی مدل بررسی کرده ایم .



حال می خواهیم برخی از feature ها را که تاثیری در یادگیری نداشته و اتفاقا می توانند دقت یادگیری را هم پایین بیاورند

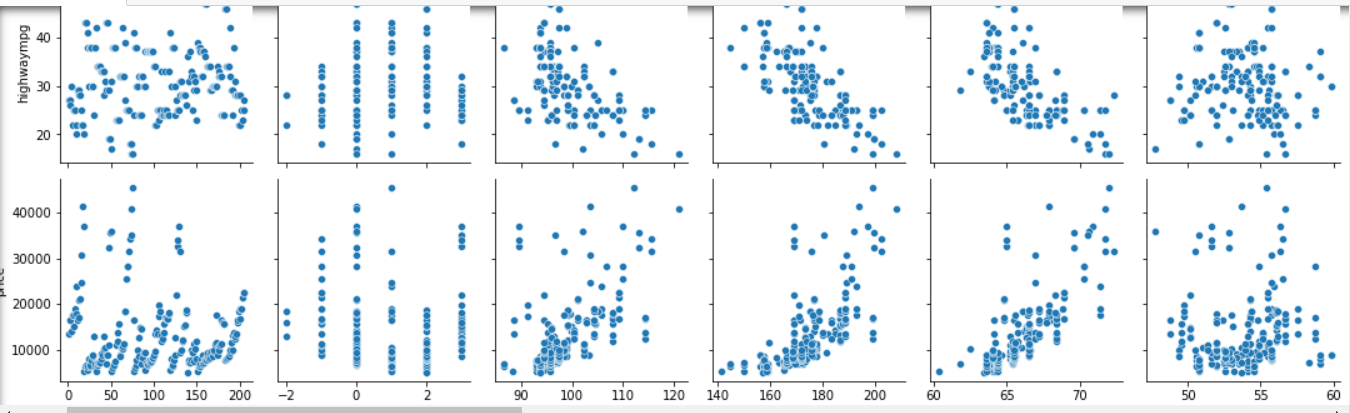
حذف کنیم .

با دستورات زیر نمودار ارتباط تمامی feature ها نسبت به هم آورده شده است

plt.figure(figsize = (30,30))

sns.pairplot(first\_df)

plt.show()

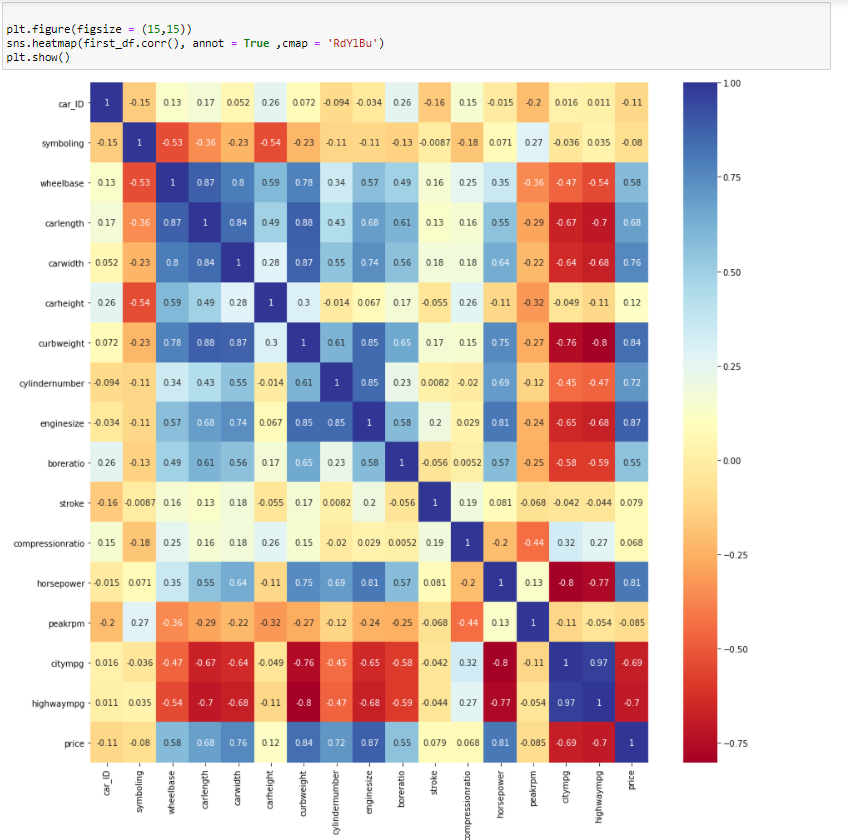


تصویر بالا نمودارهای دوبعدی تک تک feature ها را نسبت به هم دیگر نشان می دهد .

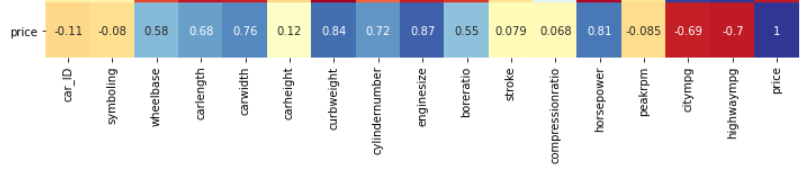
اما برای ما سطر آخر که ارتباط price را با دیگر ویژگی ها نشان می دهد مهم است ، با مشاهده ی نمودار ها می توانیم

ببینیم که بعضی از ویژگی ها مثل carheight و ... ارتباط کمی با price دارند و کاندید حذف هستند .

روش دیگر استفاده از heatmap ویژگی ها نسبت به یکدیگر می باشد .



سطر آخر که مربوط به price است را در نظر می گیریم

همانطور که انتظار داریم برخی ویژگی ها نظیر car\_id و carheight و ... ارتباط کمی با price داشته و کاندید

حذف هستند . ما تمام ویژگی هایی که عدد مربوط به ستون آن ها در سطر price از 0.5 کم تر است را حذف می کنیم .

C:\Users\Azarbadgan\Desktop\22.PNG

ویژگی های car id و symbolling قبلا از df حذف شده اند ، heat map بالایی مربوط به data set اولیه است .

در مرحله ی بعد با حذف ویژگی های نامناسب ، مجددا تابع پیاده شده را با ورودی های جدید اجرا کرده و به تقریبا

به همان خروجی های حالت قبل می رسیم . (r2\_score کم تر )

حال می خواهیم تاثیر استفاده از feature های زیاد را روی overfitting مدل بررسی کنیم

دیدیم که با کاهش ویژگی های مورد استفاده تقریبا به همان خروجی های قبلی که تعداد feature ها زیاد بود رسیدیم.

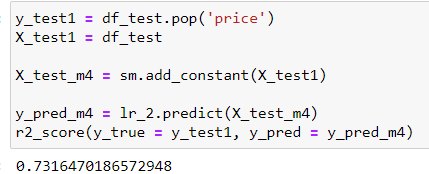
پس تفاوت کار کجاست ؟

مجدد داده ها را به دو قسمت train , test تقسیم می کنیم و الگوریتم را روی داده های train اجرا می کنیم .

دیدیم که مقدار خروجی r2\_score در این حالت حتی کمتر از حالت قبل که همه ی ویژگی ها را استفاده کرده بودیم

است ، اما تفاوت کار هنگام اجرای الگوریتم روی داده های test مشخص می شود . حالتی که تعداد feature ها زیاد بود

دیدیم که مقدار خروجی r2\_score تقریبا برابر با 0.5 بود اما با کاهش بعد داده ها به r2\_score بهتری می رسیم .



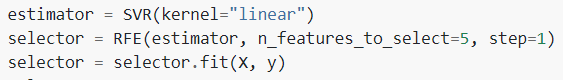
در واقع اگر چه استفاده از feature های زیاد شاید عملکرد الگوریتم را در زمان train کمی بهتر کند ( اما چون الگوریتم سعی می کند خودش را با تمامی پارامتر ها فیت کند ، عملکرد الگوریتم با پارامتر های به دست آمده از کار روی داده های train ، برای داده های test افت نسبتا زیادی می کند )

علاوه بر 1-2 روشی که برای کاهش ویژگی ها بررسی کردیم ، روش های متعدد دیگری نیز وجود دارد که هدف تمامی

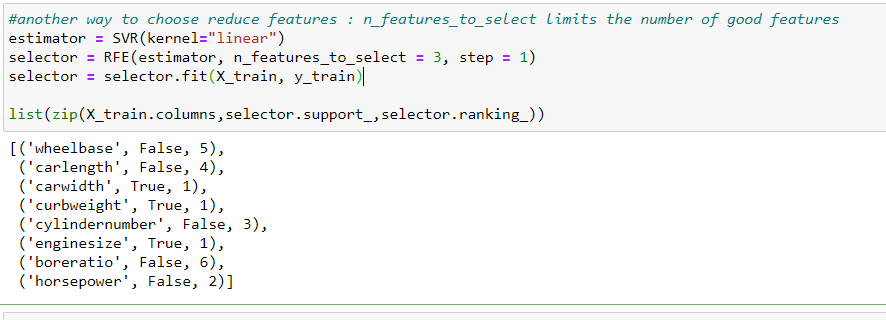
آن ها بررسی ارتباط و وابستگی بین یک ویژگی ومقدار هدف می باشد .

یک روش عملی دیگر استفاده از کلاس sklearn.feature\_selection.RFE می باشد . که در آن نوع مدل که در اینجا مدل

Linear میباشد ، تعداد feature خروجی و تعداد عملیات بازگشتی برای رسیدن به تعداد feature مد نظر.



در واقع این روش خودکار است و نیازی به انتخاب دستی توسط ما نمی باشد .

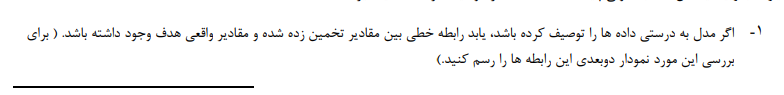


ما این روش را روی feature هایی که قبلا غربال شده بودند اجرا کردیم و این بار می خواهیم از بین این ها تنها 3 feature

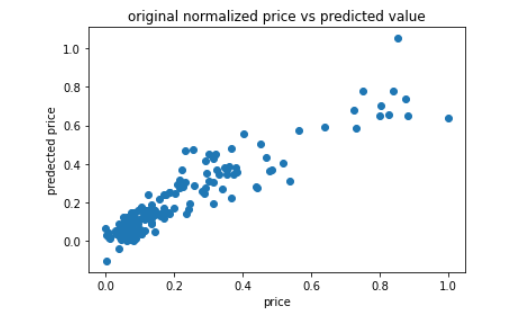
مطلوب را به ما گزارش کند و می بینیم که بهترین feature ها را enginesize , carwidth , curbweight

تشخیص داده است و پس از آن ها horsepower و ... در جایگاه های بعدی می باشند .

ارزیابی مدل رگرسیون خطی :



حین قسمت های قبل این نمودار ها را می بینیم ، نمودار ها به فرم زیر هستند

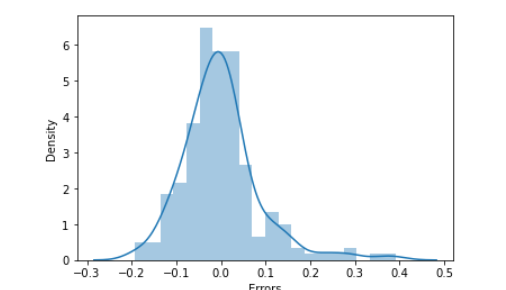


و در هر یک از قسمت ها (اعم از قسمت های با تعداد feature متفاوت ) این نمودار را هم می بینیم و محور x قیمت واقعی و

محور y قیمت پیش بینی شده است ، می بینیم که داده ها نسبت به هم نزدیک هستند و حالت خطی نسبت به هم دارند .

C:\Users\Azarbadgan\Desktop\5.PNG

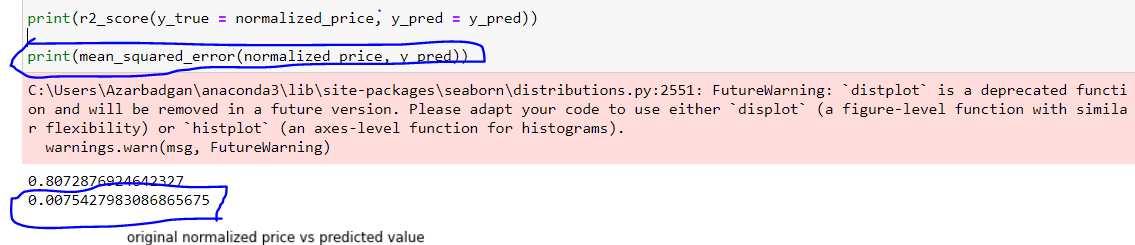
در قسمت های قبلی که تحیلیل می کردیم ، نمودار هایی به فرم زیر را می بینیم



که دارای توزیع نرمال می باشند .

C:\Users\Azarbadgan\Desktop\5.PNG

در قسمت های قبلی که تحلیل می کردیم ، دستوراتی به فرم زیر را در چند جا می بینیم :

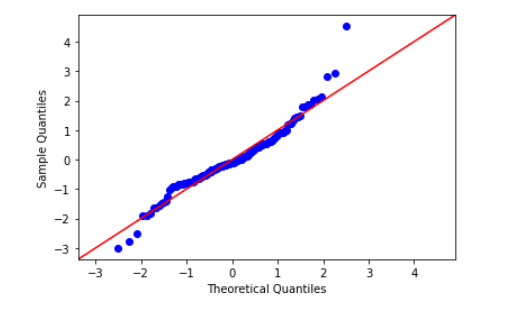


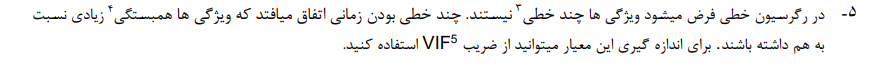
و عدد خروجی نزدیک 0 است (معمولا برای تحلیل اختلاف مقدار واقعی و مقدار پیش بینی شده از میانگین مربعات استفاده می کنیم

البته ریشه های عبارات خروجی هم میانگین خطای باقی مانده را نشان می دهد که آن هم مقار کمی می شود . )

C:\Users\Azarbadgan\Desktop\5.PNG

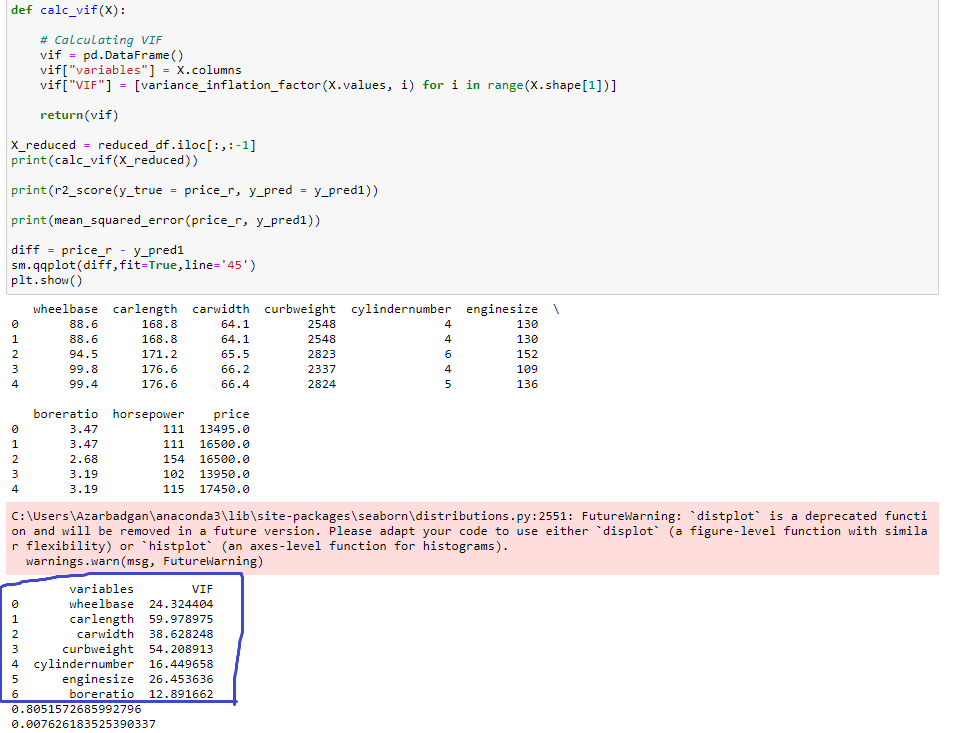
بارهم در مراحل تحلیل برای حالت های مختلفی که داریم ، نمودار هایی به فرم زیر را می بینیم :





در قسمت مر بوط به تابع پیاده سازی خودمان و با کاهش داده ها ، C:\Users\Azarbadgan\Desktop\5.PNG

Vif را حساب می کنیم و می بینیم که مقادیر vif تماما بالای 10 هستند و این خیلی بد است نشان می دهد که ویژگی ها بسیار به هم دیگر وابسته اند ؛ البته چند خطی بودن دقت الگوریتم را پایین نمی آورد اما درک ما را از تاثیر واقعی هر ویژگی روی خروجی نهایی را تحت الشعاع قرار می دهد .



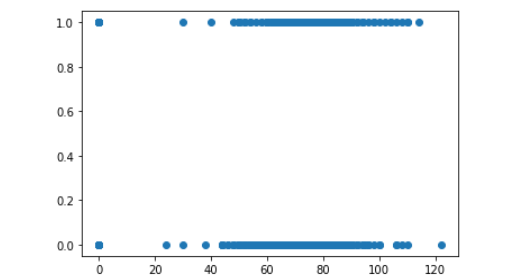
در مجموع می توان گفت که مدل پایگاه داده را خوب توصیف کرده همان طور که گفتیم ویژگی 5 ام یعنی vif تاثیر زیادی روی دقت

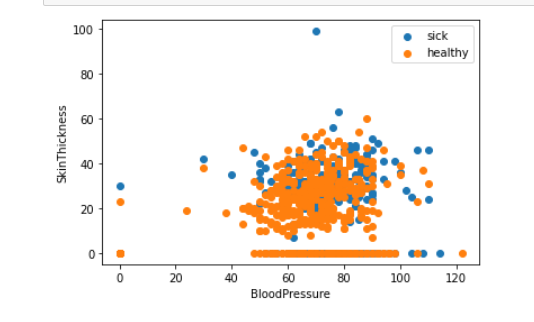
الگوریتم ندارد .

سوال2 )

در تصویر زیر می بینیم که نسبت فشار خون به نسبت خروجی کاملا نامشخص است یعنی نمی توانیم با هیچ خطی روی محور

X ها مرز درستی را مشخص کنیم .





در تصویر بالا هم می بینیم که وقتی برحسب 2 ویژگی بخواهیم یک خط را به عنوان مرز انتخاب کنیم امکان پذیر نیست پس نیاز

به feature mapping داریم و این کار در فایل ارسالی انجام شده است و این frature های جدید به جدول ویژگی اضافه شده اند .

توابع لازم برای gradient descent و cost function و نهایتا پیدا کردن پارامتر ها پیاده سازی شده اند .