به نام هستی بخش

واحد مشهد

دانشکده فنی و مهندسی، گروه کامپیوتر

«پروژه کارشناسی»

گزارش دوره Data Analysis with python

A course on cognitiveclass.ai
[1] **Powered by IBM Developer Skills Network.**

استاد راهنما :

دکتر ویسی

نگارش :

شیما شهرآئینی

پاییز 1403

مقدمه

دوره آموزشی "علم داده با پایتون" برای من یک نقطه عطف در مسیر یادگیری و رشد حرفه‌ای بود. پیش از شرکت در این دوره، با مفهوم علم داده آشنا بودم اما از ابزارها و تکنیک‌های عملی آن آگاهی چندانی نداشتم. با گذراندن این دوره، توانستم ابتدا با مفاهیم پایه مانند کتابخانه‌های ضروری پایتون و کار با داده‌ها آشنا بشوم. سپس، به تحلیل داده‌ها با استفاده از آمار توصیفی پرداخته شد که برای ارائه استدلال‌های مبتنی بر داده بسیار مفید است .در نهایت، نحوه ساخت مدل‌های یادگیری ماشین و کاربرد آن‌ها در دنیای واقعی را یادگرفتم.

این دوره آموزشی با بهره‌گیری از مجموعه داده‌ی [2] **Automobile**، به من این امکان را داد تا مفاهیم کلیدی تحلیل داده، از جمله آماده‌سازی داده(Data Wrangling)، توسعه مدل‌های مختلف (Model Development) و ارزیابی عملکرد آن‌ها (Model Evaluation) را به صورت عمیق درک کنم. با انجام پروژه‌ی عملی در محیط آزمایشگاه‌های عملی این دوره بر روی این مجموعه داده، توانستم مهارت‌های خود را در زمینه تحلیل داده با پایتون تقویت کنم.

در ادامه در طی مرور بر پروژه انجام شده بر روی این مجموعه داده به مباحث پرداخته شده در این دوره می‌پردازیم.

آماده‌سازی داده

پس از وارد کردن دیتاست مورد نظر که در فرمت csv قابل دسترسی است با استفاده از کتابخانه pandas به مرتب‌سازی و آماده‌سازی داده ها پرداختیم:

اضافه کردن header

pandas به طور خودکار header را با یک عدد صحیح از 0 تنظیم می کند. بنابراین اکنون برای توصیف بهتر داده های خود، می توانیم یک header معرفی کنیم. با توجه به اطلاعاتی که از مطالعات اولیه قبل شروع پروژه کسب کرده‌ایم **Invalid source specified.**.

شناسایی و مدیریت داده‌های بدون مقدار

1. در مجموعه داده خودرو، داده های از دست رفته با علامت سوال "؟" مشاهده می‌شوند. ما "؟" را با NaN (نه یک عدد) جایگزین می‌کنیم، که نشانگر مقدار گمشده پیش‌فرض در پایتون به دلایل سرعت و راحتی محاسباتی است. این کار را با استفاده از تابع زیر انجام دادیم که مقادیر A را با B جابه‌جا می‌کند:
   .replace(A, B, inplace = True)
2. حال برای شناسایی مقادیر از دست رفته از توابع زیر استفاده می کنیم. دو روش برای تشخیص داده های از دست رفته وجود دارد: ()isnull و ()notnull

خروجی یک مقدار بولی است که نشان می‌دهد آیا مقداری که به آرگومان ارسال می‌شود در واقع داده‌ای از دست رفته است یا خیر. سپس با استفاده از آن به شمارش مقادیر از دست رفته در هر ستون می‌پردازیم؛ با استفاده از یک حلقه for در پایتون، می توانیم به سرعت تعداد مقادیر از دست رفته در هر ستون را بفهمیم. همانطور که در بالا ذکر شد، "True" یک مقدار گم شده را نشان می دهد و "False" به معنای وجود مقدار در مجموعه داده است. در بدنه حلقه for، متد "."value_counts() تعداد مقادیر "True" را می شمارد.

```
for column in missing_data.columns.values.tolist():
    print(column)
    print (missing_data[column].value_counts())
    print("")
```

که با توجه به این کد به نتایج زیر رسیدیم:

price	peak-rpm	horsepower	stroke	bore	num-of-doors	normalized-losses	Column
4	2	2	4	4	2	41	#missing value

3. برای مدریت و مقابله با داده‌های از دست رفته میتوان 2 عملکرد را پیش بگیریم.

حذف داده‌ها:

- حذف کل سطر
- حذف کل ستون

در این روش، سطرها یا ستون‌هایی که حاوی داده‌های گمشده هستند، به طور کامل حذف می‌شوند .این روش زمانی مناسب است که تعداد داده‌های گمشده کم باشد و حذف آن‌ها تأثیر زیادی بر حجم داده‌ها نداشته باشد.

جایگزینی داده‌ها:

- جایگزینی با میانگین: جایگزینی داده‌های گمشده با میانگین مقادیر موجود در همان ستون.
- جایگزینی با فراوانی: جایگزینی داده‌های گمشده با مقدار بیشترین فراوانی در همان ستون.
- جایگزینی بر اساس توابع دیگر: استفاده از توابع آماری یا مدل‌های پیش‌بینی برای جایگزینی داده‌های گمشده.

ستون‌های تنها در صورتی حذف می‌شوند که اکثر ورودی‌های آن‌ها خالی باشند .در مجموعه داده ما، هیچ ستونی به اندازه کافی خالی نیست که بتوان آن را به طور کامل حذف کرد.

"normalized-losses"، "stroke"، "bore"، "horsepower"، "peak-rpm" را با میانگین جایگزین می‌کنیم.
جایگزینی با میانگین برای این داده‌ها می‌تواند یک روش مناسب باشد، زیرا این ویژگی‌ها معمولاً توزیع نرمال یا نزدیک به نرمال دارند. و برای این کار به طور نمونه به این صورت عمل کردیم:

```
avg_bore = df["bore"].astype("float").mean(axis=0)
df.replace({"bore": np.nan}, {"bore": avg_bore}, inplace=True)
```

"num-of-doors" را با "four" جایگزین می‌کنیم. دلیل: 84% داده‌ها از نوع چهار در هستند. از آنجایی که چهار در بیشترین فراوانی را دارد، احتمالاً این مقدار صحیح است. برای این کار به این صورت عمل کردیم:

```
df['num-of-doors'].value_counts().idxmax() #'four' is more common
```

```
'four'
```

```
df.replace({"num-of-doors": np.nan}, {"num-of-doors": "four"}, inplace=True)
```

برای "price" کل سطر حذف می‌شود. به این دلیل که "price" متغیری است که می‌خواهیم پیش‌بینی کنیم . هر ورودی داده‌ای بدون داده قیمت نمی‌تواند برای پیش‌بینی استفاده شود؛ بنابراین هر سطری که اکنون بدون داده قیمت است برای ما مفید نیست. برای این کار به این صورت عمل کردیم:

```
# simply drop whole row with NaN in "price" column
df.dropna(subset=["price"], axis=0, inplace=True)

# reset index, because we droped two rows
df.reset_index(drop=True, inplace=True)
```

تبدیل انواع داده‌ها به فرمت مناسب

آخرین مرحله در پاکسازی داده ها بررسی و اطمینان از اینکه همه داده ها در قالب صحیح هستند (int، float، text یا موارد دیگر) است. در کتابخانه pandas از ()dtype برای بررسی نوع داده و از ()atype برای تغییر نوع داده استفاده می‌کنیم.

با بررسی توع داده‌ها متوجه شدیم برخی از ستون ها از نوع داده صحیح نیستند. متغیرهای عددی باید دارای نوع 'float' یا 'int' باشند و متغیرهای دارای رشته مانند دسته‌بندی‌ها(categories) باید نوع 'object' داشته باشند. به عنوان مثال، متغیرهای 'bore' و 'stroke' مقادیر عددی هستند که موتورها را توصیف می کنند، بنابراین باید انتظار داشته باشیم که آنها از نوع 'float' یا 'int' باشند. با این حال، آنها به عنوان نوع "شی" نشان داده شده‌اند؛ ما باید انواع داده ها را با استفاده از روش "astype()" به فرمت مناسب برای هر ستون تبدیل کنیم.

```
df[["bore", "stroke"]] = df[["bore", "stroke"]].astype("float")
df[["normalized-losses"]] = df[["normalized-losses"]].astype("int")
df[["price"]] = df[["price"]].astype("float")
df[["peak-rpm"]] = df[["peak-rpm"]].astype("float")
```

استانداردسازی داده‌ها

استانداردسازی داده‌ها فرآیندی است که داده‌ها را به یک قالب مشترک تبدیل می‌کند و به محقق اجازه می‌دهد تا مقایسه‌های معناداری انجام دهد.

- استانداردسازی داده‌ها به شما کمک می‌کند تا مقایسه‌های عادلانه بین داده‌های مختلف انجام دهید.
- انتخاب واحد مناسب برای استانداردسازی داده‌ها بستگی به هدف تحلیل و نیازهای کاربردی دارد.
- در برخی موارد، ممکن است نیاز به استانداردسازی داده‌ها با استفاده از روش‌های پیچیده‌تری مانند نرمال‌سازی یا مقیاس‌بندی داده‌ها باشد.

به عنوان مثال در این پروژه ما به تبدیل mpg به L/100km پرداختیم.

در مجموعه داده ما، ستون‌های مصرف سوخت "city-mpg" و "highway-mpg" با واحد mpg (مایل در گالن) نشان داده شده‌اند .فرض کنید در حال توسعه یک برنامه در کشوری هستیم که مصرف سوخت را با استاندارد L/100km می‌پذیرد. برای تبدیل mpg به L/100km ، باید از تبدیل داده استفاده کنیم.

فرمول تبدیل: L/100km = 235.21 / mpg

```
df['city-L/100km'] = 235/df["city-mpg"]
df['highway-L/100km'] = 235/df['highway-mpg']

df.head()
```

نرمال‌سازی داده‌ها

نرمال‌سازی فرآیندی است که مقادیر چندین متغیر را به یک محدوده مشابه تبدیل می‌کند. نرمال‌سازی‌های معمول شامل مقیاس‌بندی متغیر به گونه‌ای است که میانگین متغیر صفر شود، مقیاس‌بندی متغیر به گونه‌ای که واریانس آن ۱ شود یا مقیاس‌بندی متغیر به گونه‌ای که مقادیر متغیر در محدوده ۰ تا ۱ قرار بگیرند.

- نرمال‌سازی داده‌ها در بسیاری از الگوریتم‌های یادگیری ماشین ضروری است، زیرا این الگوریتم‌ها معمولاً به داده‌هایی با مقیاس مشابه نیاز دارند.

- روش‌های مختلفی برای نرمال‌سازی داده‌ها وجود دارد، مانند نرمال‌سازی مین-مکس، استانداردسازی و نرمال‌سازی .Z-score انتخاب روش مناسب بستگی به نوع داده‌ها و الگوریتم مورد استفاده دارد.
- نرمال‌سازی داده‌ها می‌تواند به بهبود عملکرد مدل‌های یادگیری ماشین کمک کند.

در این پروژه برای نشان دادن نرمال‌سازی، فرض کردیم می‌خواهیم ستون‌های"length" ، "width"و "height" را مقیاس‌بندی کنیم. می‌خواهیم این متغیرها را به گونه‌ای نرمال‌سازی کنیم که مقادیر آن‌ها در محدوده ۰ تا ۱ قرار بگیرند.

روش : مقدار اصلی را با (مقدار اصلی)/(حداکثر مقدار) جایگزین می‌کنیم.

```
df['length'] = df['length']/df['length'].max()
df['width'] = df['width']/df['width'].max()
df['height'] = df['height']/df['height'].max()
```

گروه‌بندی(Binning)

گروه‌بندی (Binning) فرآیندی است که متغیرهای عددی پیوسته را به گروه‌های گسسته "سطل" تبدیل می‌کند تا تحلیل گروهی را ساده‌تر کند.

- تعداد سطل‌ها را با توجه به هدف تحلیل و توزیع داده‌ها انتخاب کنید.
- می‌توانید از روش‌های مختلفی برای تقسیم داده‌ها به سطل‌ها استفاده کنید، مانند روش‌های کوانتایل، فاصله مساوی و روش‌های مبتنی بر فاصله بین داده‌ها.
- گروه‌بندی بیش از حد می‌تواند باعث از دست رفتن اطلاعات شود، بنابراین باید با دقت انجام شود.

در مجموعه داده ما، "horsepower"یک متغیر با مقادیر حقیقی است که از 48 تا 288 متغیر است و 59 مقدار منحصر به فرد دارد. اگر فقط به تفاوت قیمت بین خودروهایی با اسب بخار بالا، متوسط اسب بخار و اسب بخار کم (3 نوع) اهمیت دهیم، می‌توانیم آن‌ها را برای ساده‌سازی تحلیل به سه "سطل" تقسیم کنیم!

```
bins = np.linspace(min(df["horsepower"]), max(df["horsepower"]), 4)
bins
```

```
array([ 48.        , 119.33333333, 190.66666667, 262.        ])
```

```
group_names = ['Low', 'Medium', 'High']

df['horsepower-binned'] = pd.cut(df['horsepower'], bins, labels=group_names, include_lowest=True )
df[['horsepower','horsepower-binned']].head(20)
```

متغیر نشانگر(Indicator Variable)

متغیر نشانگر یک متغیر عددی است که برای برچسب‌گذاری دسته‌ها استفاده می‌شود .آن‌ها به عنوان "dummies" نامیده می‌شوند زیرا اعداد خود به خود معنایی ندارند. از متغیرهای نشانگر برای استفاده از متغیرهای دسته‌ای در تحلیل رگرسیون در استفاده می‌شود.

- اگر یک متغیر دسته‌ای دارای بیش از دو دسته باشد، تعداد متغیرهای نشانگر ایجاد شده برابر با تعداد دسته‌ها منهای یک خواهد بود.

- در برخی موارد، ممکن است نیاز به حذف یکی از متغیرهای نشانگر برای جلوگیری از همخطی کامل بین متغیرها باشد.

در ستون "fuel-type" دو مقدار منحصر به فرد "gas" یا "diesel" وجود دارد .رگرسیون کلمات را درک نمی‌کند، فقط اعداد را درک می‌کند .برای استفاده از این ویژگی در تحلیل رگرسیون، "fuel-type"را به متغیرهای نشانگر تبدیل می‌کنیم. به عنوان نمونه به همین صورت، یک متغیر نشانگر برای ستون "aspiration" نیز ایجاد کردیم.

```
dummy_variable2 = pd.get_dummies(df["aspiration"])
dummy_variable2.rename(columns={'std':"aspiration_std", 'turbo':"aspiration_turbo"},inplace=True)

df = pd.concat([df, dummy_variable_1], axis=1)

df.drop("aspiration", axis = 1, inplace=True)
```

تحلیل ویژگی‌های داده‌ها

پیدا کردن ویژگی‌هایی که بیشترین تاثیر را بر روی متغیری که می‌خواهیم پیش‌بینی کنیم داشته باشد از اهمیت ویژه‌ای برخوردار است چراکه کمک میکند در ادامه در هنگام طراحی مدل مناسب بتوانیم بهترین عملکرد را داشته باشیم و در انجام پروژه هم در نهایت ویژگی‌های مهم موثر را پیدا کردیم:

تحلیل الگوهای ویژگی‌ها با استفاده از تجسم داده (Analyzing Individual Feature Patterns Using Visualization)

هنگام (Data Visualization) تجسم متغیرهای فردی، مهم است که ابتدا بفهمیم با چه نوع متغیری سروکار دارید. این به ما کمک می‌کند تا روش تصویرسازی مناسب را برای آن متغیر پیدا کنیم.

- **متغیرهای عددی پیوسته**

متغیرهای عددی پیوسته متغیرهایی هستند که ممکن است حاوی هر مقداری در محدوده ای باشند. آنها می توانند از نوع "int64" یا "float64" باشند. یک راه عالی برای تجسم این متغیرها استفاده از نمودارهای پراکنده با خطوط متناسب است. به منظور شروع درک رابطه (خطی) بین یک متغیر منفرد و قیمت، می‌توانیم از "regplot" استفاده کنیم که نمودار پراکندگی به‌علاوه خط رگرسیون مناسب برای داده‌ها را ترسیم می‌کند. به عنوان نمونه این 3 مثال از ویژگی‌های بررسی شده:

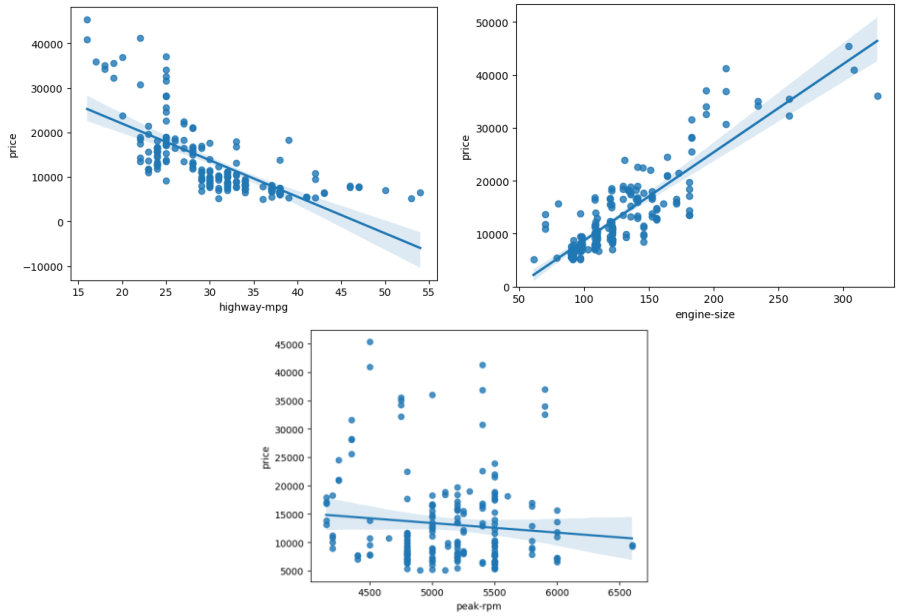
```
#scatterplot of "engine-size" and "price".
#Positive Linear Relationship

sns.regplot(x="engine-size", y="price", data=df)
plt.ylim(0,)
```

روش ()corr. در پانداس یک ابزار قدرتمند برای محاسبه ضریب همبستگی بین متغیرهای عددی در یک DataFrame است.

ضریب همبستگی یک عدد بین 1- تا 1 است که نشان‌دهنده میزان و جهت ارتباط خطی بین دو متغیر است:

- **ضریب 1 :** نشان‌دهنده همبستگی کامل مثبت است، یعنی با افزایش یک متغیر، متغیر دیگر نیز افزایش می‌یابد.
- **ضریب 1-: ** نشان‌دهنده همبستگی کامل منفی است، یعنی با افزایش یک متغیر، متغیر دیگر کاهش می‌یابد.
- **ضریب 0 :** نشان‌دهنده عدم وجود همبستگی بین دو متغیر است.

روش corr(). به طور پیش‌فرض از ضریب همبستگی پیرسون استفاده می‌کند که برای اندازه‌گیری ارتباط خطی بین متغیرهای پیوسته استفاده می‌شود .با این حال، می‌توان از انواع دیگر ضریب همبستگی مانند اسپیرمن و کندال نیز استفاده کرد.

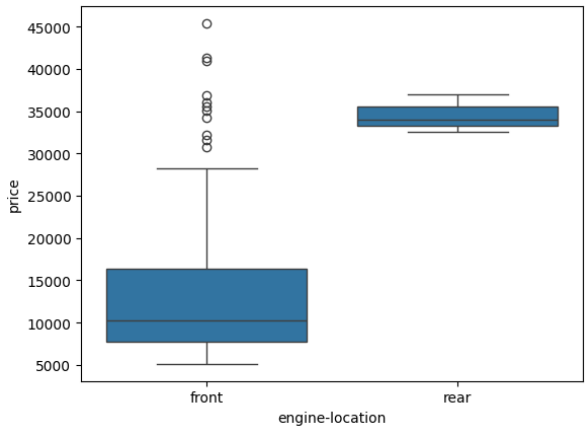


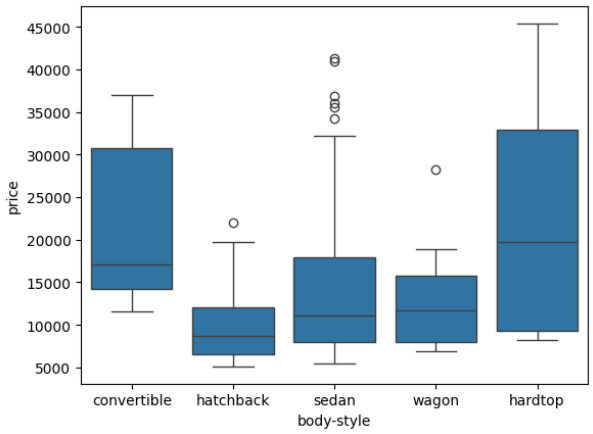
ویژگی	"engine-size"	"highway-mpg"	"peak-rpm"
تجسم	صعودی	نزولی	خط تقریبا افقی
همبستگی	0.872335	-0.704692	-0.101616
پیش‌بینی کننده مناسب قیمت	است	است	نیست

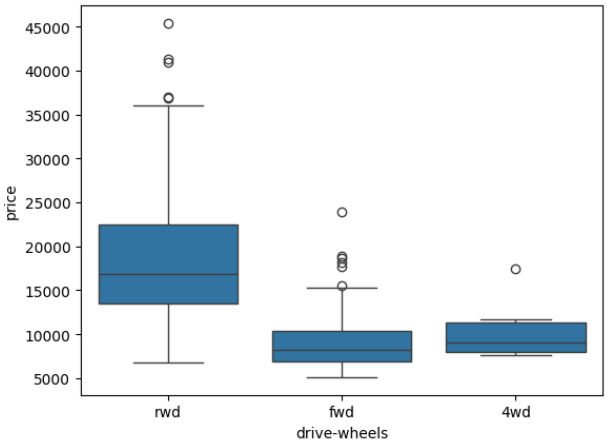
- **متغیرهای دسته بندی شده**

اینها متغیرهایی هستند که "ویژگی" یک واحد داده را توصیف می کنند و از گروه کوچکی از دسته ها انتخاب می شوند. متغیرهای طبقه‌بندی می‌توانند نوع "object" یا "int64" داشته باشند. یک راه خوب برای تجسم متغیرهای طبقه‌بندی، استفاده از نمودارهای جعبه‌ای(boxplots) است. . به عنوان نمونه این 3 مثال از ویژگی‌های بررسی شده:

```
#We see that the distributions of price between the different body-style categories have a significant overlap,
#so body-style would not be a good predictor of price.

sns.boxplot(x="body-style", y="price", data=df)
```







توزیع قیمت بین دسته‌های مختلف "body-style" همپوشانی قابل توجهی دارند، بنابراین "body-style" ممکن است پیش‌بینی کننده خوبی برای قیمت نباشد.

توزیع قیمت بین دسته‌های "engine-location" متفاوت است، بنابراین "engine-location" می‌تواند یک پیش‌بینی کننده بالقوه برای قیمت باشد.

توزیع قیمت بین دسته‌های "drive-wheels" نیز متفاوت است، بنابراین "drive-wheels" نیز می‌تواند یک پیش‌بینی کننده بالقوه برای قیمت باشد.

تحلیل آماری توصیفی (Descriptive Statistical Analysis)

درک اولیه از داده‌ها : اولین قدم در هر تحلیل داده، درک ماهیت داده‌ها است. تحلیل توصیفی به ما کمک می‌کند تا بدانیم داده‌های ما چه شکلی هستند، چه توزیعی دارند و چه ویژگی‌هایی دارند.

شناسایی الگوها و روندها : با استفاده از تحلیل توصیفی می‌توانیم الگوها و روندهای پنهان در داده‌ها را شناسایی کنیم.

آماده‌سازی داده‌ها برای تحلیل‌های بعدی : تحلیل توصیفی به ما کمک می‌کند تا داده‌های خود را تمیز کنیم، خطاها را برطرف کنیم و برای تحلیل‌های پیچیده‌تر آماده کنیم.

- **تابع describe**

به طور خودکار آمارهای پایه را برای همه متغیرهای پیوسته محاسبه می‌کند .هر مقدار NaN به طور خودکار در این آمارها حذف می‌شود.

این تابع نشان می‌دهد: تعداد متغیرها، میانگین، انحراف استاندارد(std) ، حداقل مقدار، ، حداکثر مقدار، دامنه بین‌چرکی (IQR: 25%, 50% and 75%)

```
#df.describe()
df.describe(include=['object'])
```

	make	num-of-doors	body-style	drive-wheels	engine-location	engine-type	num-of-cylinders	fuel-system
count	201	201	201	201	201	201	201	201
unique	22	2	5	3	2	6	7	8
top	toyota	four	sedan	fwd	front	ohc	four	mpfi
freq	32	115	94	118	198	145	157	92

- **تابع Value Counts**

روش value_counts یک روش مناسب برای درک تعداد واحدهای هر ویژگی یا متغیر است. روش value_counts فقط روی سری‌های pandas کار می‌کند، نه روی داده فریم‌های pandas. بنابراین، فقط از یک براکت df['drive-wheels'] استفاده می‌کنیم، نه دو براکت df[['drive-wheels']].

```
# engine location would not be a good predictor variable for the price.
#This is because we only have three cars with a rear engine and 198 with an engine in the front, so this result is skewed.
#Thus, we are not able to draw any conclusions about the engine location

engine_loc_counts = df['engine-location'].value_counts().to_frame()
engine_loc_counts.rename(columns={'engine-location': 'value_counts'}, inplace=True)
engine_loc_counts.index.name = 'engine-location'
engine_loc_counts.head(10)
```

	count
engine-location	
front	198
rear	3

مبانی گروه‌بندی (Basics of Grouping)

روش groupby داده‌ها را بر اساس دسته‌های مختلف گروه‌بندی می‌کند .داده‌ها بر اساس یک یا چند متغیر گروه‌بندی می‌شوند و تحلیل روی هر گروه به صورت جداگانه انجام می‌شود. با استفاده از روش groupby می‌توانید داده‌ها را بر اساس ویژگی‌های مختلف گروه‌بندی کنیم و تحلیل‌های عمیق‌تری انجام دهیم. به عنوان نمونه از استفاده آن در پروژه داریم:

```
#it seems rear-wheel drive vehicles are, on average, the most expensive,
#while 4-wheel and front-wheel are approximately the same in price.

df_group_one = df[['drive-wheels','body-style','price']]

# grouping results
df_group_one_test_one = df_group_one.groupby(['drive-wheels'], as_index=False)['price'].mean()
df_group_one_test_one
```

	drive-wheels	price
0	4wd	10241.000000
1	fwd	9244.779661
2	rwd	19757.613333

از داده‌های ما، به نظر می‌رسد خودروهای با سیستم محرک عقب، به طور متوسط، گران‌ترین هستند، در حالی که سیستم‌های محرک چهار چرخ و جلو تقریباً قیمت یکسانی دارند.

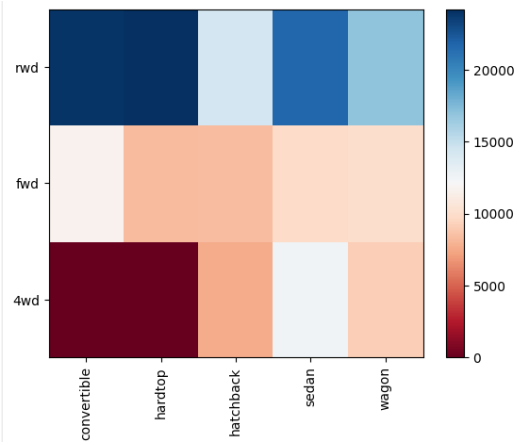
این داده‌های گروه‌بندی شده هنگام تبدیل به یک **جدول محوری(pivot table)**، بسیار آسان‌تر قابل تجسم هستند .جدول محوری مانند یک صفحه گسترده اکسل است، با یک متغیر در امتداد ستون و دیگری در امتداد سطر .می‌توانیم از روش "pivot"برای تبدیل داده‌فریم به یک جدول محوری استفاده کنیم.

پس از آن میتوان با تجسم آن بهتر نتایج را درک که می‌توان از **نمودار نقشه حرارتی (Heatmap)** یک روش بصری برای نمایش ماتریس همبستگی بین متغیرها استفاده کرد. در این نمودار، رنگ هر سلول نشان‌دهنده شدت همبستگی بین دو متغیر است.

به عنوان نمونه:

```
#pivot table
grouped_pivot = df_group_one_test_two.pivot(index='drive-wheels',columns='body-style')
grouped_pivot = grouped_pivot.fillna(0) #fill missing values with 0
grouped_pivot
```

	price				
body-style	**convertible**	**hardtop**	**hatchback**	**sedan**	**wagon**
drive-wheels					
4wd	0.0	0.000000	7603.000000	12647.333333	9095.750000
fwd	11595.0	8249.000000	8396.387755	9811.800000	9997.333333
rwd	23949.6	24202.714286	14337.777778	21711.833333	16994.222222



P-value

به همبستگی(correlation) پیش‌تر پرداختیم حال لازم است با P-value بپردازیم:

مقدار P احتمال آن است که همبستگی بین دو متغیر از نظر آماری معنی‌دار باشد .معمولاً سطح معنی‌داری 0.05 را انتخاب می‌کنیم، به این معنی که 95٪ مطمئن هستیم که همبستگی بین متغیرها معنی‌دار است.

به طور قراردادی، زمانی که:

- مقدار P کمتر از 0.001 است : می‌گویم شواهد قوی وجود دارد که همبستگی معنی‌دار است.
- مقدار P کمتر از 0.05 است : شواهد متوسطی وجود دارد که همبستگی معنی‌دار است.

- مقدار P کمتر از 0.1 است : شواهد ضعیفی وجود دارد که همبستگی معنی‌دار است.
- مقدار P بیشتر از 0.1 است : هیچ مدرکی وجود ندارد که همبستگی معنی‌دار باشد.

در این پروژه ما تمام ویژگی‌های دیتاست را بررسی کردیم مانند نمونه:

```
#Pearson Correlation Coefficient and P-value of 'wheel-base' and 'price'

pearson_coef, p_value = stats.pearsonr(df['wheel-base'], df['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p_value)

#Since the p-value is $<$ 0.001, the correlation between wheel-base and price is statistically significant,
#although the linear relationship isn't extremely strong (~0.585).
```

تحلیل واریانس(ANOVA)

تحلیل واریانس (ANOVA) یک روش آماری است که برای بررسی اینکه آیا تفاوت‌های معنی‌داری بین میانگین‌های دو یا چند گروه وجود دارد، استفاده می‌شود. ANOVA دو پارامتر را برمی‌گرداند:

- F-test score: ANOVA فرض می‌کند که میانگین همه گروه‌ها یکسان است، میزان انحراف میانگین‌های واقعی از این فرض را محاسبه می‌کند و آن را به عنوان نمره آزمون F گزارش می‌دهد .نمره بالاتر به معنای تفاوت بزرگ‌تر بین میانگین‌ها است.
- P-value : مقدار P نشان می‌دهد که نمره محاسبه شده ما چقدر از نظر آماری معنی‌دار است.

اگر متغیر قیمت ما با متغیری که در حال تجزیه و تحلیل آن هستیم به شدت همبسته باشد، انتظار داریم ANOVA یک نمره آزمون F قابل توجه و یک مقدار P کوچک را برگرداند.

```
df_gptest = df[['drive-wheels', 'price']]
grouped_test2=df_gptest.groupby(['drive-wheels'])
```

```
# ANOVA
f_val, p_val = stats.f_oneway(grouped_test2.get_group(('fwd',))['price'], grouped_test2.get_group(('rwd',))['price'], grouped_t

print( "ANOVA results: F=", f_val, ", P =", p_val)
```

```
ANOVA results: F= 67.95406500780399 , P = 3.3945443577149576e-23
```

This is a great result with a large F-test score showing a strong correlation and a P-value of almost 0 implying almost certain statistical significance. But does this mean all three tested groups are all this highly correlated?

نتایج تحلیل ویژگی‌های داده‌ها

اکنون می‌دانیم که داده‌هایمان چگونه به نظر می‌رسند و چه متغیرهایی برای پیش‌بینی قیمت خودرو مهم هستند. ما آن را به متغیرهای زیر محدود کرده ایم:

Length، Width، Curb-weight، Engine-size، Horsepower، City-mpg، Highway-mpg، Wheel-base، Bore، Drive-wheels

همانطور که اکنون به سمت ساخت مدل‌های یادگیری ماشینی برای خودکارسازی تحلیل خود حرکت می‌کنیم، تغذیه مدل با متغیرهایی که به طور معناداری بر متغیر هدف ما تأثیر می‌گذارند، عملکرد پیش‌بینی مدل ما را بهبود می‌بخشد.

توسعه مدل

بعد از آشنایی کامل به دیتاست و آماده‌سازی آن به دنبال بهترین مدل برای آن می‌گردیم که به بهترین شکل fit دیتاهای ما شود. در این دوره به ساده ترین مدل‌ها پرداخته شد.

Linear Regression

مدل خطی ساده یک روش برای درک رابطه بین دو متغیر است:

- متغیر پیش‌بینی‌کننده یا مستقل(X)
- متغیر پاسخ یا وابسته(Y)

نتیجه رگرسیون خطی یک توابع خطی است که متغیر پاسخ (وابسته) را به عنوان تابعی از متغیر پیش‌بینی‌کننده (مستقل) پیش‌بینی می‌کند.

توابع خطی: Yhat = a + b*X

a: عرض از مبدا خط رگرسیون، به عبارت دیگر :مقدار Y وقتی X برابر با 0 است.
b: شیب خط رگرسیون، به عبارت دیگر :مقداری که Y با افزایش X به اندازه 1 واحد تغییر می‌کند

```
from sklearn.linear_model import LinearRegression

# import the visualization package: seaborn
import seaborn as sns
```

```
lm = LinearRegression()
lm
```

```
LinearRegression
LinearRegression()
```

```
#how highway-mpg can help us predict car price.

X = df[['highway-mpg']]
Y = df['price']

lm.fit(X,Y)
Yhat = lm.predict(X)
```

وقتی صحبت از رگرسیون خطی ساده می شود، یک راه عالی برای تجسم برازش مدل با استفاده از "regression plots" است. که پیش‌تر به آن اشاره شد.

```
plt.figure(figsize=(width, height))
sns.regplot(x="highway-mpg", y="price", data=df)
```

نمودار باقیمانده(Residual Plot)

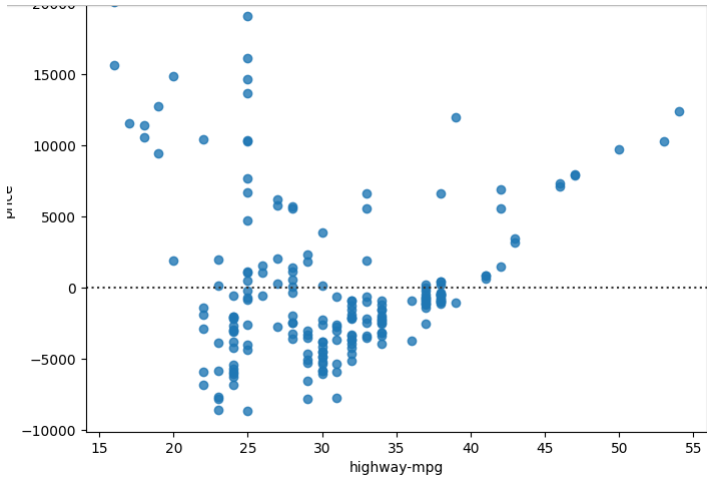
نمودار باقیمانده یک روش خوب برای تجسم واریانس داده‌ها است. تفاوت بین مقدار مشاهده شده (y) و مقدار پیش‌بینی شده (Yhat) را باقیمانده (e) می‌نامند. وقتی به یک نمودار رگرسیون نگاه می‌کنیم، باقیمانده فاصله از نقطه داده تا خط رگرسیون برازش شده است.

```
plt.figure(figsize=(width, height))
sns.residplot(x=df['highway-mpg'],y=df['price'])
plt.show()
```

در این نمودار به پراکندگی باقیمانده‌ها توجه می‌کنیم:

اگر نقاط در نمودار باقیمانده به طور تصادفی در اطراف محور x پخش شده باشند، یک مدل خطی برای داده‌ها مناسب است.

و این به این دلیل است که پراکندگی تصادفی باقیمانده‌ها به معنای ثابت بودن واریانس است و بنابراین مدل خطی برای این داده‌ها مناسب است.



Multiple Linear Regression

رگرسیون خطی چندگانه بسیار شبیه به رگرسیون خطی ساده است، اما این روش برای توضیح رابطه بین یک متغیر پاسخ (وابسته) پیوسته و دو یا چند متغیر پیش‌بینی‌کننده (مستقل) استفاده می‌شود.

بیشتر مدل‌های رگرسیون در دنیای واقعی شامل چندین پیش‌بینی‌کننده هستند. در این پروژه ساختار را با استفاده از چهار متغیر پیش‌بینی‌کننده نشان دادیم، اما این نتایج را می‌توان به هر عدد صحیح تعمیم داد:

Yhat = a + b1*X1* + *b2*X2 + b3*X3* + *b4*X4 + ...

```
Z = df[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']]

lm3 = LinearRegression ()

lm3.fit(Z, df['price'])

Y_hat3 = lm3.predict(Z)

print( "intercept = ", lm3.intercept_, "slope ", lm3.coef_)
```

تجسم یک مدل برای رگرسیون خطی چندگانه کمی پیچیده‌تر می‌شود زیرا نمی‌توان آن را با رگرسیون یا نمودار باقیمانده تجسم کرد. یک راه برای بررسی برازش مدل، مشاهده نمودار توزیع است. می توان به توزیع مقادیر برازش حاصل از مدل نگاه کرد و آن را با توزیع مقادیر واقعی مقایسه کرد.

```
plt.figure(figsize=(width, height))

ax1 = sns.kdeplot(df['price'], color="r", label="Actual Value")
sns.kdeplot(Y_hat3, color="b",  label="Fitted Values" , ax=ax1)
```

Polynomial Regression

رگرسیون چندجمله‌ای یک مورد خاص از مدل رگرسیون خطی عمومی یا مدل‌های رگرسیون خطی چندگانه است.

با توان دوم یا بالاتر متغیرهای پیش‌بینی‌کننده، روابط غیرخطی به دست می‌آوریم.

انواع مختلف رگرسیون چندجمله‌ای:

Quadratic - 2nd Order: Yhat = a + b_1 X +b_2 X^2

Cubic - 3rd Order: Yhat = a + b_1 X +b_2 X^2 +b_3 X^3

Higher-Order: Y = a + b_1 X +b_2 X^2 +b_3 X^3 ....

```
#We saw earlier that a linear model did not provide the best fit while using "highway-mpg" as the predictor variable.
#Let's see if we can try fitting a polynomial model to the data instead.

x = df['highway-mpg']
y = df['price']

# Here we use a polynomial of the 3rd order (cubic)
f = np.polyfit(x, y, 3)
p = np.poly1d(f)  #display the polynomial function
print(p)
```

```
        3         2
-1.557 x + 204.8 x - 8965 x + 1.379e+05
```

برای تجسم یک مدل برای رگرسیون چندجمله‌ای تابعی نوشتیم:

```
#function to plot the polynomial data:
def PlotPolly(model, independent_variable, dependent_variabble, Name):
    x_new = np.linspace(15, 55, 100)
    y_new = model(x_new)

    plt.plot(independent_variable, dependent_variabble, '.', x_new, y_new, '-')
    plt.title('Polynomial Fit with Matplotlib for Price ~ Length')
    ax = plt.gca()
    ax.set_facecolor((0.898, 0.898, 0.898))
    fig = plt.gcf()
    plt.xlabel(Name)
    plt.ylabel('Price of Cars')

    plt.show()
    plt.close()
```

در این دوره دیدیم برای چندجمله‌ای‌های با درجه بالاتر و تعداد متغیرهای بیشتر، عبارت پیچیده‌تر می‌شود به همین دلیل در نهایت بهترین روش اعمال این مدل به صورت زیر است:

```
from sklearn.preprocessing import PolynomialFeatures
```

```
pr=PolynomialFeatures(degree=2)
pr
```

```
PolynomialFeatures()
```

```
Z.shape
```

```
(201, 4)
```

```
Z_pr=pr.fit_transform(Z)
Z_pr.shape
```

```
(201, 15)
```

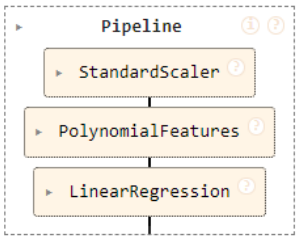
Pipeline

یک روش کارآمد برای پردازش داده‌ها به صورت مرحله به مرحله است. تصور کنید یک کارخانه تولید دارید. هر مرحله از تولید یک محصول، از مواد اولیه تا محصول نهایی، توسط یک ماشین انجام می‌شود. خط لوله داده هم به همین شکل عمل می‌کند. هر مرحله از پردازش داده‌ها توسط یک بخش خاص از خط لوله انجام می‌شود.

- **سادگی :** به جای نوشتن کدهای جداگانه برای هر مرحله، همه مراحل را در یک خط لوله قرار می‌دهیم .
- **کارایی :** خط لوله‌ها معمولاً به صورت موازی اجرا می‌شوند که باعث افزایش سرعت پردازش می‌شود .
- **تکرارپذیری :** یک خط لوله را می‌توان بارها و بارها اجرا کرد بدون اینکه نیاز به تغییر کد باشد.

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
```

```
Input=[('scale',StandardScaler()), ('polynomial', PolynomialFeatures(include_bias=False)), ('model',LinearRegression())]

pipe = Pipeline(Input)
pipe
```



ارزیابی مدل‌ها

هنگام ارزیابی مدل‌های خود، نه تنها می‌خواهیم نتایج را تجسم کنیم، بلکه همچنین به یک معیار کمی برای تعیین دقت مدل نیاز داریم.

دو معیار بسیار مهم که اغلب در آمار برای تعیین دقت یک مدل استفاده می‌شوند عبارتند از:

- **R^2 / R-squared**

R squared، همچنین به عنوان ضریب تعیین شناخته می‌شود، یک معیار برای نشان دادن میزان نزدیکی داده‌ها به خط رگرسیون برازش شده است.
مقدار R-squared نشان‌دهنده درصد تغییرات متغیر پاسخ (y) است که توسط یک مدل خطی توضیح داده می‌شود.

- **میانگین مربعات خطا(MSE)**

MSE میانگین مربعات خطاها را اندازه‌گیری می‌کند .یعنی تفاوت بین مقدار واقعی (y) و مقدار برآورد شده.(ŷ)

به عنوان نمونه در پروژه داریم:

```
print('The R-square is: ', lm.score(X, Y))

#We can say that \~49.659% of the variation of the price is explained by this simple linear model,
#which we have built using the highway_mpg data.
```

```
The R-square is:  0.4965911884339176
```

```
mse = mean_squared_error(df['price'], Yhat)
print('The mean square error of price and predicted value is: ', mse)
```

```
The mean square error of price and predicted value is:  31635042.944639888
```

هنگام مقایسه مدل‌ها، مدلی با مقدار R-squared بالاتر و مقدار MSE کوچک‌تر ، برازش بهتری برای داده‌ها دارد.

بیایید به مقادیر برای مدل‌های مختلف نگاه کنیم:

- **رگرسیون خطی ساده :** استفاده از Highway-mpg به عنوان متغیر پیش‌بینی‌کننده قیمت:
  - R-squared: 0.49659118843391759
  - MSE: 3.16 x 10^7
- **رگرسیون خطی چندگانه :** استفاده از Horsepower ، Curb-weight، Engine-sizeو Highway-mpg به عنوان متغیرهای پیش‌بینی‌کننده قیمت:
  - R-squared: 0.80896354913783497
  - MSE: 1.2 x 10^7
- **برازش چندجمله‌ای :**استفاده از Highway-mpg به عنوان متغیر پیش‌بینی‌کننده قیمت:
  - R-squared: 0.6741946663906514
  - MSE: 2.05 x 10^7

تجزیه و تحلیل:

- بر اساس مقادیر R-squared ، مدل رگرسیون خطی چندگانه بهترین برازش را برای داده‌ها دارد، زیرا مقدار R-squared آن بالاترین است.
- بر اساس مقادیرMSE ، مدل رگرسیون خطی چندگانه نیز بهترین برازش را دارد، زیرا مقدار MSE آن کوچک‌ترین است.

این نتایج نشان می‌دهند که استفاده از چندین متغیر پیش‌بینی‌کننده می‌تواند به بهبود دقت مدل پیش‌بینی قیمت خودرو کمک کند.

ارزیابی و اصلاح مدل (Model Evaluation and Refinement)

آموزش و آزمون

یک گام مهم در آزمایش مدل، تقسیم داده ها به داده های آموزشی و آزمایشی است.

```
y_data = df['price']
x_data = df.drop('price',axis=1)

x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.10, random_state=1)
```

Cross-validation یک روش آماری برای ارزیابی عملکرد مدل‌های یادگیری ماشین است. این روش به ما کمک می‌کند تا از overfitting جلوگیری کنیم و عملکرد مدل را به درستی ارزیابی کنیم.

Cross-validation ' : مجموعه آموزش' و 'مجموعه تست داده‌ها را به دو بخش تقسیم می‌کند. مدل بر روی مجموعه آموزش آموزش داده می‌شود و عملکرد آن بر روی مجموعه تست ارزیابی می‌شود. این فرآیند به ما کمک می‌کند تا عملکرد مدل را بر روی داده‌های جدید و نادیده ارزیابی کنیم.

که از انواع آن، در این پروژه با K-Fold Cross-Validation آشنا شدیم. K-Fold Cross-Validation داده‌ها به K بخش مساوی تقسیم می‌شوند. در هر تکرار، یکی از بخش‌ها به عنوان مجموعه تست و بقیه به عنوان مجموعه آموزش استفاده می‌شوند. این فرآیند K بار تکرار می‌شود و میانگین نتایج به عنوان عملکرد نهایی مدل در نظر گرفته می‌شود.

```
yhat_ = cross_val_predict(lre,x_data[['horsepower']], y_data,cv=4)
yhat_[0:5]
```

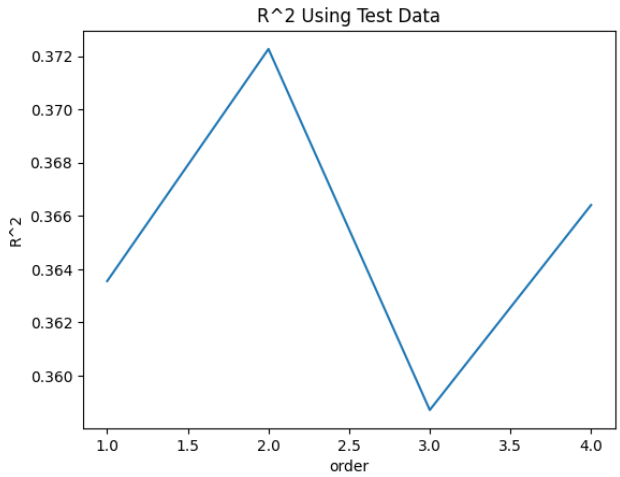
Overfitting

زمانی رخ می‌دهد که مدل به جای فرایند اصلی، مدل با نویز مطابقت داشته باشد .بنابراین، هنگام آزمایش مدل با استفاده از مجموعه آزمایشی، به خوبی عمل نمی‌کند زیرا در حال مدل‌سازی نویز است، نه فرایند اصلی که رابطه را ایجاد کرده است.

R^2منفی نشانه‌ای از overfitting است.

انتخاب مرتبه مناسب

ما در این پروژه متوجه اهمیت مرتبه مدل چند جمله ای شدیم و دیدیم که R^2 با افزایش مرتبه به تدریج افزایش می یابد تا زمانی که از یک چند جمله‌ای مرتبه سه استفاده شود. سپس، R^2 به طور چشمگیری در چند جمله ای مرتبه چهار کاهش می یابد.



Ridge Regression

یک روش رگرسیونی است که برای جلوگیری از overfitting استفاده می‌شود .در این روش، یک اصطلاح جریمه به تابع هزینه اضافه می‌شود که باعث می‌شود ضرایب مدل به سمت صفر تمایل پیدا کنند .این کار باعث می‌شود مدل ساده‌تر شده و احتمال بیش‌برازش کاهش یابد.

پارامتر آلفا میزان تأثیر اصطلاح جریمه را کنترل می‌کند .اگر مقدار آلفا صفر باشد، رگرسیون ریج به رگرسیون خطی معمولی تبدیل می‌شود .با افزایش مقدار آلفا، ضرایب مدل به شدت به سمت صفر میل می‌کنند و مدل ساده‌تر می‌شود.

لازم به ذکر است که استفاده از داده‌های تست به عنوان داده‌های اعتبارسنجی در این مثال برای ارزیابی مدل در طول فرایند تنظیم پارامترها است.

```
RigeModel=Ridge(alpha=1)

RigeModel.fit(x_train_pr, y_train)

yhat_ridge = RigeModel.predict(x_test_pr)
```

Grid search

یک روش exhaustively برای جستجوی بهترین ترکیب از هایپرپارامترها است. به صورت جامع تمامی ترکیب‌های ممکن از پارامترهای مشخص شده را امتحان می‌کند تا بهترین عملکرد مدل را پیدا کند.

در این پروژه اصطلاح آلفا یک هایپرپارامتر است. Sklearn دارای کلاس GridSearchCV است تا فرآیند یافتن بهترین هایپرپارامتر را ساده‌تر کند.

```
from sklearn.model_selection import GridSearchCV
```

```
parameters1= [{'alpha': [0.001,0.1,1, 10, 100, 1000, 10000, 100000, 100000]}]
RR=Ridge()
Grid1 = GridSearchCV(RR, parameters1,cv=4)

Grid1.fit(x_data[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']], y_data)

BestRR=Grid1.best_estimator_

BestRR.score(x_test[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']], y_test)
```

```
0.6373011574906102
```

References

[1] "https://courses.cognitiveclass.ai/certificates/dc8757b72ca340baafa1574a75d0c5ee," [Online].

[2] "https://archive.ics.uci.edu/dataset/10/automobile," [Online].