**بسمه تعالی**

دانشگاه علامه طباطبائی

**پروژه درس رگرسیون ۱**

**عنوان:**

# بررسی مدل رگرسیون خطی ساده و مدل رگرسیون خطی چندگانه برای داده‌های kc_house_data

**استاد: دکتر پورطاهری**

**تنظیم کننده: علی شکارچی**

## مقدمه:

در فایل **kc_house_data** داده‌هایی در رابطه با املاک مسکونی موجود است است که این داده‌ها به بررسی ۱۸ متغیر برای منازل مسکونی پرداخته است.

در این پروژه با انتخاب متغیر پاسخ از بین این متغیرها:

در بخش ۱ با انتخاب یکی دیگر از متغیر ها که بهترین همبستگی را با متغیر پاسخ دارد، یک مدل رگرسیون خطی ساده بر داده‌ها برازش داده میشود و ابعاد مختلف آن نظیر نمودار پراکنش، ضرایب رگرسیون، فواصل اطمینان برای ضرایب، آزمون ضرایب، نمودارهای احتمال نرمال و پراکنش مانده‌ها بررسی میشود.

و در بخش ۲ با انتخاب چند متغیر دیگر با روش گام به گام یک مدل رگرسیون خطی چندگانه بر داده‌ها برازش داده میشود و ابعاد مختلف آن نظیر آزمون ضرایب مدل و ضریب تعیین و هم‌خطی بررسی میشود و مدل رگرسیون ریج بر داده‌ها برازش داده میشود.

لازم به ذکر است در انجام این پروژه از زبان برنامه نویسی پایتون استفاده شده و در تمام مراحل قطعه کدهای مربوطه پیوست و توضیح داده شده است. (کل فایل کد بصورت جدا پیوست شده است.)

## بخش اول (رگرسیون خطی ساده):

**در گام اول اطلاعات متغیر ها که در فایل CSV در ۱۸ ستون موجود است، با زبان برنامه نویسی پایتون بازخوانی شده و در ۱۸ لیست از اعداد ذخیره شده است. (این قسمت در فایل کد اصلی قابل ملاحظه است.)**

### انتخاب متغیر پاسخ:

**با توجه به اطلاعات فایل در بین متغیرهای موجود بنظر میرسد برای اطلاعات املاک مسکونی متغیر قیمت خانه متغیر پاسخ برای مدل‌های رگرسیونی است.**

### انتخاب متغیر مستقل:

**در این مرحله مبنای انتخاب متغیر مستقل بیشترین همبستگی با متغیر پاسخ است. پس همبستگی برای متغیرهای مستقل و پاسخ دو به دو بررسی میشود و متغیری که بیشترین همبستگی را با متغیر پاسخ داشته باشد برای مدل رگرسیون خطی ساده انتخاب میشود. مطابق با قطعه کد زیر متغیر sqft_livin (مساحت پذیرایی یعنی x3) بعنوان متغیر مستقل با ضریب همبستگی ۰/۷۰۴ انتخاب میشود.**

```
independent_variables = [\
x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, x16, x17]
corrcoeff = 0
index = 0
for i in independent_variables:
    tmp = abs(np.corrcoef(y, i)[0, 1])
    if tmp > corrcoeff:
        corrcoeff = tmp
        index = independent_variables.index(i)+1
corrcoeff_result = (round(corrcoeff,2), index)
```

**Out:**

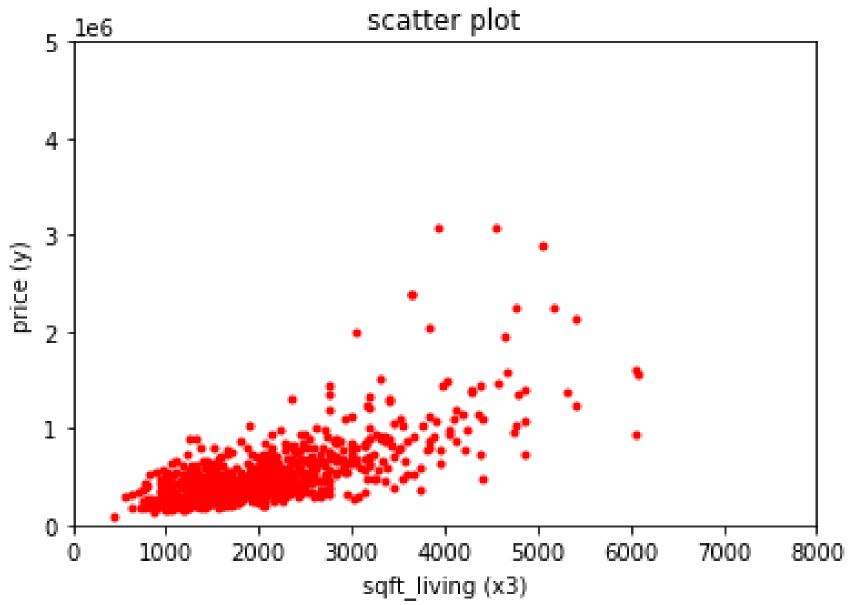
```
In [2]: corrcoeff_result
Out[2]: (0.7, 3)
```

## سوال ۱: (نمودار پراکنش **y** در مقابل **x** و نتایج حاصل)

**مطابق با قطعه کد زیر نمودار زیر برای متغیر y و x3 رسم میشود.**

```
plt.scatter(x3, y, color="r", marker=".")
plt.title("scatter plot")
plt.xlabel("sqft_living (x3)")
plt.ylabel("price (y)")
plt.xlim(0,8000)
plt.ylim(0,5000000)
plt.show()
```

**Out:**




**مشاهده میشود که رابطه خطی خوبی بین y و x3 بویژه در بازه $x3 \in$ (۵۰۰.۳۵۰۰) و $y \in$ (۱۰۰۰۰۰.۱۵۰۰۰۰۰) برقرار است بنحوی که با افزایش مقدار x3 مقدار y نیز افزایش پیدا میکند یا بعبارتی قیمت خانه با مساحت پذیرایی رابطه مستقیمی دارد. همچنین مشاهده میشود که با دور شدن x3 از ۳۵۰۰ و y از ۱۵۰۰۰۰۰ پراکندگی دادهها افزایش پیدا میکند و رابطه خطی ضعیفتر میشود.**

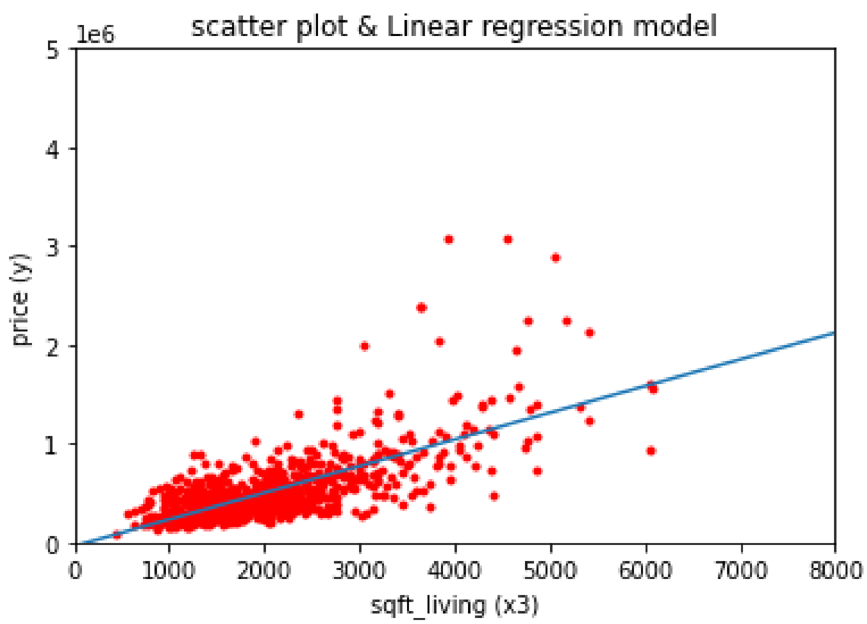
## سوال ۲: (برازش مدل رگرسیون خطی و براورد پارمترهای $\beta_0$ و $\beta_1$)

مطابق قطعه کد زیر آماره‌های $S_{xy}$ و $S_{xx}$ و سپس با استفاده از روابط ضرایب رگرسیون محاسبه و مدل خطی برازش داده میشود.

```
n = len(y)
xbar = np.mean(x3)
ybar = np.mean(y)
sumxiyi = sum(xi*yi for xi, yi in zip(x3, y))
sumxi2 = sum(xi**2 for xi in x3)
sxy = sumxiyi - n*xbar*ybar
sxx = sumxi2 - n*xbar**2
β1hat = sxy / sxx
β0hat = ybar - β1hat*xbar
regression_model_result = ("%.5f x %.5f" % (β1hat,β0hat))
```

**out:**

```
In [10]: sxy
Out[10]: 171818607859.1123

In [11]: sxx
Out[11]: 637008235.1871667

In [12]: regression_model_result
Out[12]: '269.72745 x -34626.60622'
```



## سوال ۳: (فواصل اطمینان ۹۵٪ برای $\beta_۰$ و $\beta_۱$)

**با استفاده از اطلاعات سوالات قبل مطابق قطعه کد زیر آماره‌های sst و ssr و sse و سپس $se\widehat{\beta}_۱$ و $se\widehat{\beta}_۰$ محاسبه و فواصل اطمینان $\beta_۰$ و $\beta_۱$ با فرض و $t_{\frac{\alpha}{2},(n-2)} = z_{\frac{\alpha}{2}} = ۱/۹۶$ تشکیل داده میشود.**

```
sst = sumyi2 = sum(yi**2 for yi in y)
ssr = β1hat*sxx
sse = sst - ssr
mse = sse / n-2
seβ1hat = (mse/sxx) ** 0.5
tα2 = zα2 = 1.96
β1_confidence_interval = (β1hat-seβ1hat*tα2, β1hat+seβ1hat*tα2)
seβ0hat = (mse**0.5) * ((1/n)+(xbar**2/sxx))**0.5
β0_confidence_interval = (β0hat-seβ0hat*tα2, β0hat+seβ0hat*tα2)
```

**Out:**

```
In [15]: ssr
Out[15]: 171818607859.1123

In [16]: sse
Out[16]: 148466336781496.6

In [17]: seβ1hat
Out[17]: 17.651868120645826

In [18]: β1_confidence_interval
Out[18]: (235.12978997971194, 304.32511301264356)

In [19]: seβ0hat
Out[19]: 40161.232303185956

In [20]: β0_confidence_interval
Out[20]: (-113342.62153001215, 44089.4090984768)
```

**سوال ۴: (آزمون فرض t–استیودنت** $\begin{cases} H_{\cdot}: \beta_{۱} = \cdot \\ H_{۱}: \beta_{۱} \neq \cdot \end{cases}$ **در سطح معنی داری ۵٪):**

**با استفاده از اطلاعات سوالات قبل مطابق قطعه کد زیر آماره T محاسبه و با مقایسه قدر مطلق آن با مقدار** $t_{\frac{\alpha}{2}.(n-2)} = z_{\frac{\alpha}{2}} = ۱/۹۶$ **محاسبه میشود و نتیجه آزمون رد فرض** $H_{\cdot}$ **با قاطعیت و پذیرش مدل رگرسیون میباشد.**

```
T = (β1hat-0)/seβ1hat
t_assumption_result = abs(T) > tα2
```

**Out:**

```
In [23]: T
Out[23]: 15.280391268089152

In [24]: t_assumption_result
Out[24]: True
```

## سوال ۵: (آزمون فرض $H_0: \beta_1 = 0$ با استفاده از جدول تجزیه واریانس در سطح معنی داری ۵٪)

**با استفاده از اطلاعات سوالات قبل مطابق قطعه کد زیر جدول تجزیه واریانس که در زیر آمده تشکیل داده میشود و با مقایسه $F_0 = \frac{MSR}{MSE}$ و $F_{\alpha,1,(n-2)}$ فرض $H_0$ با قاطعیت رد میشود و مدل رگرسیون مورد پذیرش میباشد.**

```
from scipy.stats import f
dfr = 1
msr = ssr/dfr
F0 = msr/mse
Fα = f.ppf(q=0.05, dfn=1, dfd=n)
f_assumption_result = F0 > Fα
```

**Out:**

```
In [44]: msr
Out[44]: 171818607859.1123

In [45]: F0
Out[45]: 0.8656529248718458

In [46]: Fα
Out[46]: 0.003934779661818523

In [47]: f_assumption_result
Out[47]: True
```

| sov | df | SS | MS |
|---|---|---|---|
| regression | ۱ | $SSR = ۱۷۱۸۱۸۶۰۷۸۵۹/۱۱۲۳$ | $MSR = \frac{SSR}{۱} = ۱۷۱۸۱۸۶۰۷۸۵۹/۱۱۲۳$ |
| error | $n - ۲$ | $SSE = ۱۴۸۴۶۶۳۳۶۷۸۱۴۹۶/۶$ | $MSE = \frac{SSE}{n - ۲} = ۱۹۸۴۸۴۴۰۷۴۵۹/۸۹۳۸$ |
| total | $n - ۱$ | $SST = ۱۴۸۶۳۸۱۵۵۳۸۹۳۵۵/۷۲$ | – |

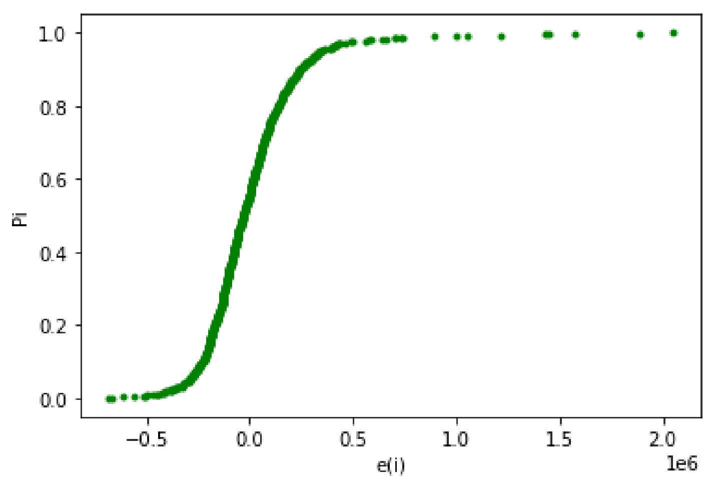
$$F_0 = \frac{MSR}{MSE} = ۰/۸۶۵۶۵ > ۰/۰۰۳۹۳ = F_{\alpha,1,(n-2)}$$

**سوال ۶: (نمودار احتمال نرمال و بررسی فرض نرمال بودن توزیع احتمال خطاها)**

**با استفاده از اطلاعات سوالات قبل و مطابق قطعه کد زیر خطاها با $y_i - \hat{y}_i$ برای n داده محاسبه و سپس مرتب و احتمالات با $p_i = \frac{i-\frac{1}{2}}{n}$ برای n داده محاسبه و نمودار آن رسم میشود.**

```
error = [y[i] - (β1hat*x3[i] + β0hat) for i in range(n)]
error.sort()
P = [(i-0.5)/n for i in range(1,n+1)]
plt.plot(error,P, "g.")
plt.xlabel("e(i)")
plt.ylabel("Pi")
plt.show()
```

**Out:**



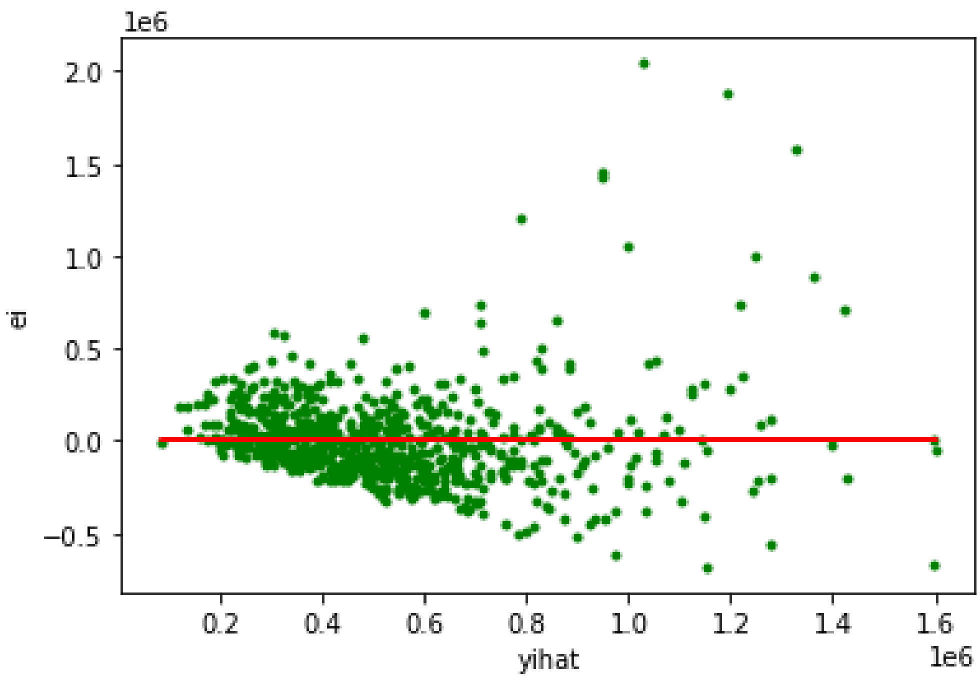

**فرض نرمال بودن توزیع احتمال خطاها با توجه به خطی بودن نمودار حاصل برای نقاط میانی توزیع پذیرفته میشود ولی ملاحظه میشود که دمها انحراف دارند بدین صورت که دمهای توزیع احتمالات خطاها به مراتب سنگین‌تر از دمهای توزیع احتمال نرمال هستند یا بعبارتی بالا تر از دمهای توزیع نرمال واقع میشوند.**

## سوال ۷: (نمودار پراکنش مانده‌ها در برابر $\widehat{y}_i$ و بررسی آن)

**با استفاده از اطلاعات سوالات قبل و مطابق قطعه کد زیر مانده‌ها با $y_i - \widehat{y}_i$ برای n داده محاسبه و نمودار پراکنش مانده‌ها در برابر $\widehat{y}_i$ و $error = ۰$ رسم میشود.**

```
yhat = [(β1hat*x3[i] + β0hat) for i in range(n)]
error = [y[i] - yhat[i] for i in range(n)]
plt.plot(yhat, error, "g.")
plt.plot(yhat, [0 for i in range(n)], color="r")
plt.xlabel("yihat")
plt.ylabel("ei")
plt.show()
```

**out:**




**الف) فرض خطی بودن مدل:**

**مطابق با نمودار بالا با کمی اغماض و با صرف نظر از داده‌های پرت میتوان پراکندگی مانده‌ها را نسبت به صفر متقارن در نظر گرفت که تاییدکننده فرض خطی بودن مدل است.**

**ب) فرض ثابت بودن واریانس:**

**مطابق با نمودار بالا ملاحظه میشود که با افزایش $\widehat{y_i}$ پراکندگی مانده‌ها افزایش پیدا میکند که فرض ثابت بودن واریانس را رد میکند.**

**پ) فرض وجود داده‌های پرت:**

**مطابق با نمودار بالا وجود داده‌های پرت تایید میشود که یا ناشی از خطای اندازه‌گیری و یا مربوط به تاثیر متغیرهای دیگر است که در این مدل لحاظ نشده‌اند. با توجه به اینکه افزایش قیمت خانه به عوامل متخلفی به جز مساحت بستگی دارد حالت دوم بیشتر مورد تایید است.**

**ت) ارائه تبدیل مناسب برای خطی کردن مدل:**

**همانطور که در قسمت الف به آن اشاره شد با صرف نظر از داده‌های پرت وجود رابطه خطی با اغماض مورد تایید است و تبدیل برای خطی کردن مدل در اینجا ضرورتی ندارد.**

**ث) ارائه تبدیل باکس-کاکس برای ثابت کردن واریانس:**

**تبدیل باکس-کاکس بصورت زیر در نظر گرفته میشود:**

$$v_i = \begin{cases} ({y_i}^{\lambda} - 1)/(\lambda \dot{y}^{\lambda-1}) & \lambda \neq 0 \\ \dot{y} ln(y_i) & \lambda = 0 \end{cases}$$

**که در آن $\dot{y} = (y_۱ y_۲ \cdots y_n)^{\frac{۱}{n}}$ همان میانگین هندسی $y_i$ ها است.**

با استفاده از اطلاعات سوالات قبل و مطابق قطعه کد زیر که در آن از ماژول **scipy** استفاده شده تبدیل باکس-کاکس انجام میشود. (حجم محاسبات $v_i$ ها که وابسته به مقدار $\dot{y}$ است بسیار بزرگ بوده و خارج از توان محاسبات مستقیم با ضابطه بالا است.)

```
from scipy.stats import boxcox
W, λ = boxcox(y)
```

**Out:**

```
In [21]: λ
Out[21]: -0.31259396305099313
```

(لازم به ذکر است در قطعه کد فوق تبدیل باکس-کاکس روی دادههای لیست $y$ (متغیر پاسخ) انجام شده و $\lambda$ مشخص شده و مقادیر جدید در لیست $W$ تخصیص داده شدند.)

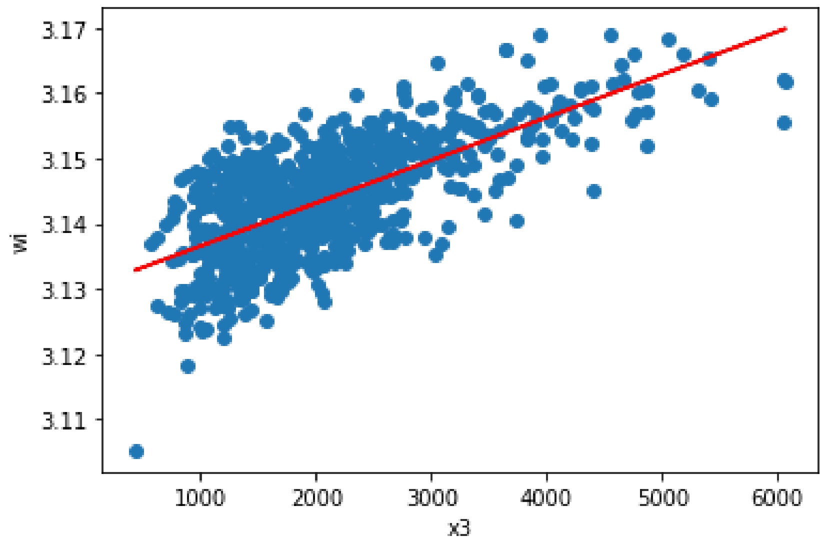
**ج) ارائه مدل نهایی:**

**پس از تبدیل باکس-کاکس برای مقادیر جدید متغیر پاسخ ($w_i$) و متغیر مستقل x3 مدل رگرسیون خطی ساده دوباره برازش داده میشود که با استفاده از اطلاعات سوالات قبل و مطابق قطعه کد زیر ضرایب رگرسیون محاسبه و نمودار پراکنش و خط برازش داده شده رسم میشود.**

```
Wbar = np.mean(W)
Wsumxiwi = sum(xi*wi for xi, wi in zip(x3, W))
Wsxy = Wsumxiwi - n*xbar*Wbar
Wβ1hat = Wsxy / sxx
Wβ0hat = Wbar - Wβ1hat*xbar
What = Wβ1hat*np.array(x3) + Wβ0hat
plt.scatter(x3, W)
plt.plot(x3, What, "r")
plt.xlabel("x3")
plt.ylabel("wi")
plt.show()
```

**Out:**

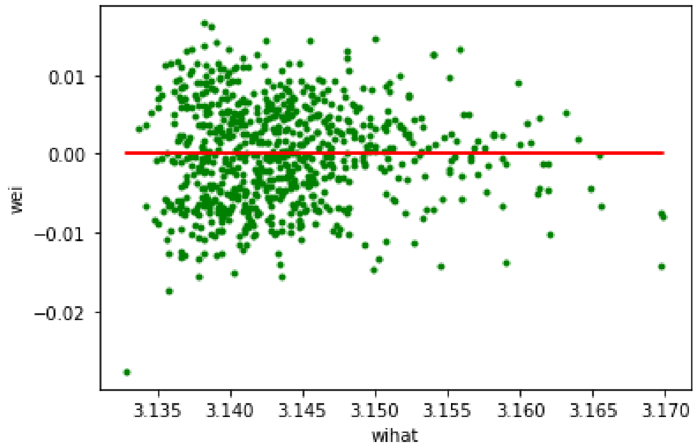
```
In [3]: Wsxy
Out[3]: 4186.968276633881

In [4]: Wβ1hat
Out[4]: 6.572863654429302e-06

In [5]: Wβ0hat
Out[5]: 3.1299953112225216

In [6]: Wregression_model_result
Out[6]: '0.0000065729 x 3.1299953112'
```




**همچنین در بررسی نمودار پراکنش مانده‌ها در برابر $\hat{W}_i$ ملاحظه میشود بین مانده‌ها و $\hat{W}_i$ الگو خاصی برقرار نیست و مانده ها پراکندگی ثابتی دارند که این نتیجه تبدیل انجام شده است. مطابق قطعه کد زیر نمودار پراکنش مانده‌ها رسم میشود.**

```
Werror = [W[i] - What[i] for i in range(n)]
plt.plot(What, Werror, "g.")
plt.plot(What, [0 for i in range(n)], color="r")
plt.xlabel("wihat")
plt.ylabel("wei")
plt.show()
```

**Out:**



0.01
0.00
−0.01
−0.02
wei
3.135
3.140
3.145
3.150
3.155
3.160
3.165
3.170
wihat