IIT ROORKEE

FREE ONLINE EDUCATION swayam शिक्षित भारत, उन्नत भारत

NPTEL ONLINE CERTIFICATION COURSE

Estimation, Prediction of Regression Model Residual Analysis: Validating Model Assumptions - I

Dr. A. Ramesh

DEPARTMENT OF MANAGEMENT STUDIES

Agenda

- Point Estimation
- Interval Estimation
- Confidence Interval for the Mean Value of *y*
- Prediction Interval for an Individual Value of *y*

Problem

- Data were collected from a sample of 10 Ice cream vendors located near college campuses.
- For the i^{th} observation or restaurant in the sample, x_i is the size of the student population (in thousands) and y_i is the quarterly sales (in thousands of dollars).
- The values of x_i and y_i for the 10 restaurants in the sample are summarized in Table

Data

Restaurant	Student Population **(1000)**	Sales **(1000)**
1	2	58
2	6	105
3	8	88
4	8	118
5	12	117
6	16	137
7	20	157
8	20	169
9	22	149
10	26	202

Python code for scatter plot

In [4]:
```
import pandas as pd
import matplotlib as mpl
import statsmodels.formula.api as sm
from sklearn.linear_model import LinearRegression
from scipy import stats
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

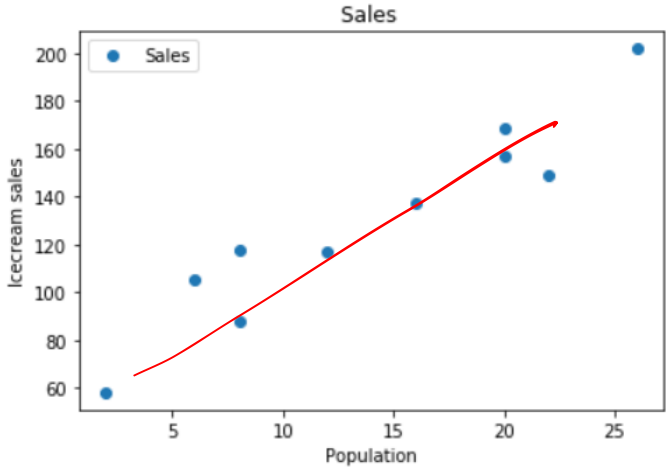
In [5]:
```
data = pd.read_excel('C:/Users/Somi/Documents/lrm.xlsx')
data
```

Out[5]:

	Restaurant	Student Population	Sales
0	1	2	58
1	2	6	105
2	3	8	88
3	4	8	118
4	5	12	117
5	6	16	137
6	7	20	157
7	8	20	169
8	9	22	149
9	10	26	202

Python code for scatter plot

```
In [36]: data.plot('Population', 'Sales', style='o')
         plt.ylabel('Icecream sales')
         plt.title('Sales ')
         plt.show()
```



Python code for regression Equation

In [37]:
```
import statsmodels.api as s
St_pop = data['Population']
sales = data['Sales']
st_pop = s.add_constant(St_pop)
model1 = sm.OLS(sales,st_pop)
result1 = model1.fit()
print(result1.summary())
```

OLS Regression Results

Dep. Variable:	Sales	R-squared:	0.903
Model:	OLS	Adj. R-squared:	0.891
Method:	Least Squares	F-statistic:	74.25
Date:	Wed, 04 Sep 2019	Prob (F-statistic):	2.55e-05
Time:	14:33:11	Log-Likelihood:	-39.342
No. Observations:	10	AIC:	82.68
Df Residuals:	8	BIC:	83.29
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	60.0000	9.226	6.503	0.000	38.725	81.275
Population	5.0000	0.580	8.617	0.000	3.662	6.338

Omnibus:	0.928	Durbin-Watson:	3.224
Prob(Omnibus):	0.629	Jarque-Bera (JB):	0.616
Skew:	-0.060	Prob(JB):	0.735
Kurtosis:	1.790	Cond. No.	33.6

Sales = 60 + 5 St-Pop

Python code for regression Equation

```
In [10]: from sklearn.linear_model import LinearRegression
```

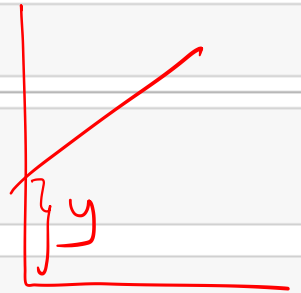
```
In [12]: x = data['Population'].values.reshape(-1,1)
         y = data['Sales'].values.reshape(-1,1)
```

```
In [13]: reg= LinearRegression()
         reg.fit(x,y)
```

```
Out[13]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

```
In [14]: reg.intercept_[0], reg.coef_[0][0]
```

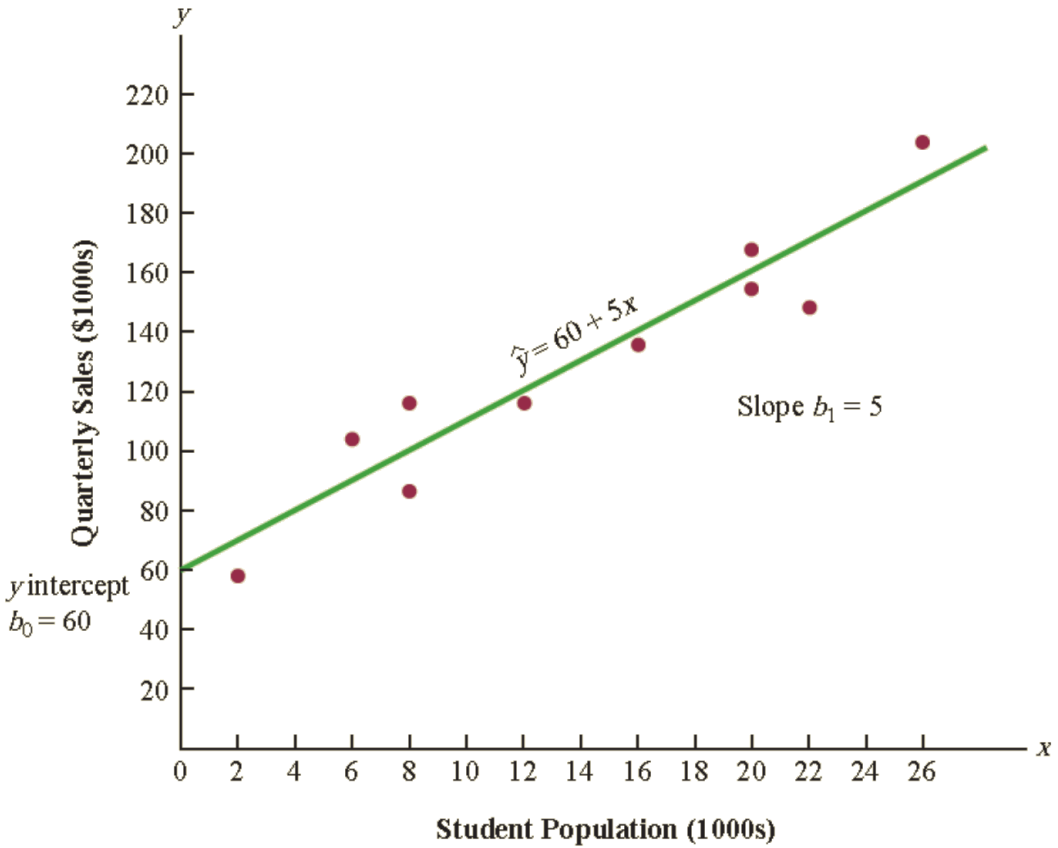
```
Out[14]: (60.0, 5.0)
```



Python code for regression

- In the Ice cream vendor example, the estimated regression equation 60 + 5*x* provides an estimate of the relationship between the size of the student population *x* and quarterly sales *y*.

$$\hat{y} = 60 + 5x$$


y
220
200
180
160
140
120
100
80
60
40
20
Quarterly Sales ($1000s)
$\hat{y} = 60 + 5x$
Slope $b_1 = 5$
y intercept
$b_0 = 60$
0 2 4 6 8 10 12 14 16 18 20 22 24 26
x
Student Population (1000s)

Point Estimate

- We can use the estimated regression equation to develop a point estimate of the mean value of *y* for a particular value of *x* or to predict an individual value of *y* corresponding to a given value of *x*.
- For instance, suppose a manager want a point estimate of the mean quarterly sales for all restaurants located near college campuses with 10,000 students.

Point estimate

- Using the estimated regression equation 60 +5*x*, we see that for *x* 10 (or 10,000 students), 60 + 5(10) = 110.
- Thus, a point estimate of the mean quarterly sales for all restaurants located near campuses with 10,000 students is $110,000.

```
In [32]: reg.predict(10)

Out[32]: array([[110.]])
```

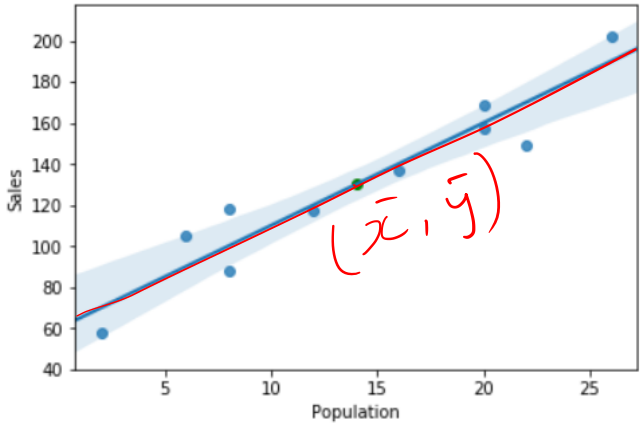
Point estimate

- Now suppose the manager want to predict sales for an individual restaurant located near College, with 10,000 students.
- In this case we are not interested in the mean value for all restaurants located near campuses with 10,000 students;
- We are just interested in predicting quarterly sales for one individual restaurant.
- As it turns out, the point estimate for an individual value of *y* is the same as the point estimate for the mean value of *y*.
- Hence, we would predict quarterly sales of 60 + 5(10) = 110 or $110,000 for this one restaurant.

Plot at mean value of x and y

In [16]:
```
x = data['Population']
y = data['Sales']
plt.figure()
sns.regplot(x,y,fit_reg= True)
plt.scatter(np.mean(x), np.mean(y), color = "green")
```

Out[16]: <matplotlib.collections.PathCollection at 0x28356eb64a8>



Confidence Interval Estimation

$E(y) = a + bx$

- **Confidence interval**, is an interval estimate of the *mean value of y* for a given value of *x*.
- **Prediction interval**, is used whenever we want an interval estimate of an *individual value of y* for a given value of *x*.
- The point estimate of the mean value of *y* is the same as the point estimate of an individual value of *y*.
- The margin of error is larger for a prediction interval.

Confidence Interval Estimation

x_p = the particular or given value of the independent variable x

y_p = the value of the dependent variable y corresponding to the given x_p

$E(y_p)$ = the mean or expected value of the dependent variable 'y' corresponding to the given x_p

$\hat{y} = b_0 + b_1 x_p$ = the point estimate of $E(y_p)$ when $x = x_p$

$60 + 5(10) = 110.$

Confidence Interval Estimation

In general, we cannot expect $\hat{y}_p$ to equal $E(y_p)$ exactly.

If we want to make an inference about how close $\hat{y}_p$ is to the true mean value $E(y_p)$, we will have to estimate the variance of $\hat{y}_p$.

The formula for estimating the variance of $\hat{y}_p$ given x_p, denoted by , is $s^2_{\hat{y}_p}$

$$s^2_{\hat{y}_p} = s^2\left[\frac{1}{n} + \frac{(x_p - \bar{x})^2}{\Sigma(x_i - \bar{x})^2}\right]$$

Confidence Interval Estimation

$\bar{X} \pm z\frac{\sigma}{\sqrt{n}}$

CONFIDENCE INTERVAL FOR $E(y_p)$

$$\hat{y}_p \pm t_{\alpha/2} s_{\hat{y}_p}$$

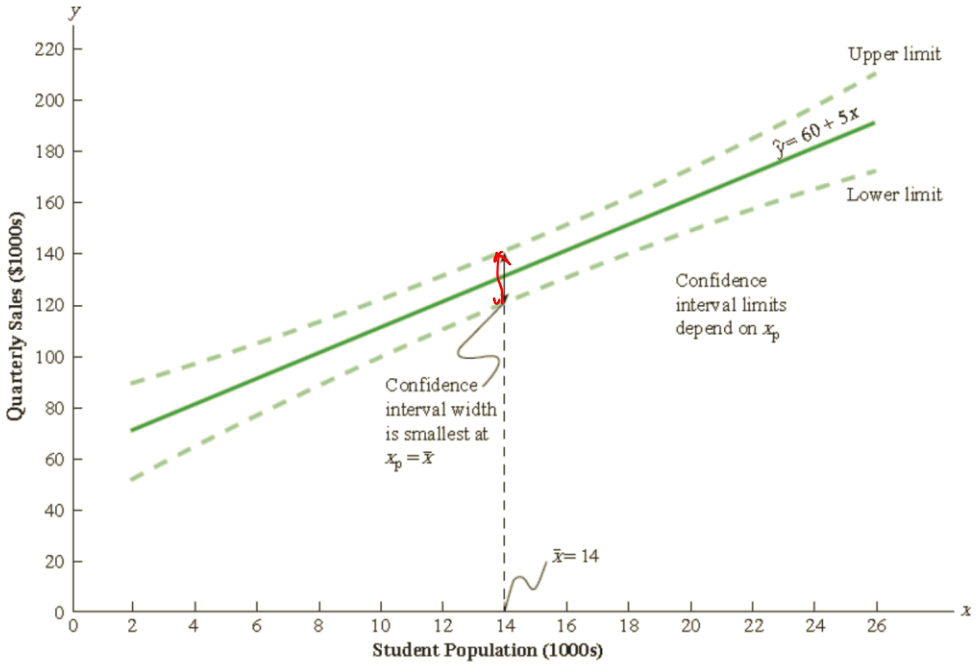
$$s^2_{\hat{y}_p} = s^2\left[\frac{1}{n} + \frac{(x_p - \bar{x})^2}{\Sigma(x_i - \bar{x})^2}\right]$$

$$s_{\hat{y}_p} = 13.829\sqrt{\frac{1}{10} + \frac{(10 - 14)^2}{568}}$$

$$= 13.829\sqrt{.1282} = 4.95$$

$$110 \pm 11.415$$

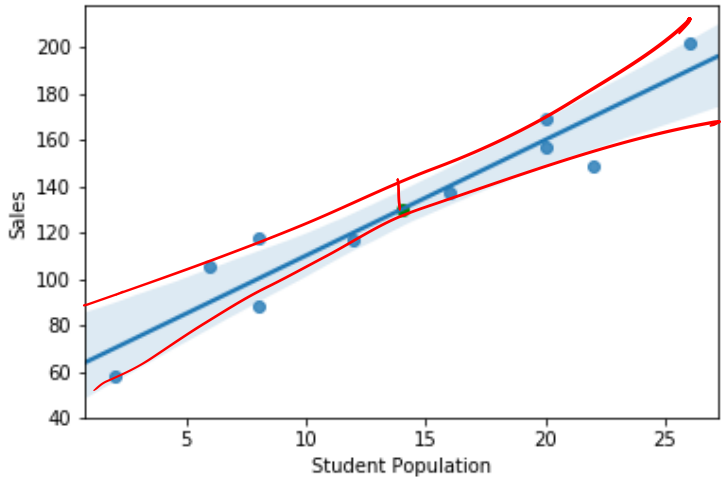
Confidence Intervals for the Mean sales *y* at given values of student population *x*



Python Code

```
In [39]: x = data['Student Population']
         y = data['Sales']
         plt.figure()
         sns.regplot(x,y,fit_reg= True)
         plt.scatter(np.mean(x), np.mean(y), color = "green")
```

```
Out[39]: <matplotlib.collections.PathCollection at 0x1fe5e250a90>
```



Special Case

The estimated standard deviation of $\hat{y}_p$

is smallest when $x_p = \bar{x}$ and the quantity $x_p - \bar{x} = 0$

$$s_{\hat{y}_p} = s\sqrt{\frac{1}{n} + \frac{(\bar{x} - x)^2}{\Sigma(x_i - \bar{x})^2}} = s\sqrt{\frac{1}{n}}$$

Prediction Interval for an Individual Value of *y*

- Instead of estimating the mean value of sales for all restaurants located near campuses with 10,000 students, we want to estimate the sales for an individual restaurant located near a particular College with 10,000 students.

(1) The variance of individual '*y*' values about the mean $E(y_p)$, an estimate of which is given by s^2

(2) The variance associated with using $\hat{y}_p$ estimate $E(y_p)$, an estimate of which is given by $s^2_{\hat{y}_p}$

Prediction Interval for an Individual Value of *y*

$$
\begin{aligned}
s^2_{\text{ind}} &= s^2 + s^2_{\hat{y}_p} \\
&= s^2 + s^2\left[\frac{1}{n} + \frac{(x_p - \bar{x})^2}{\Sigma(x_i - \bar{x})^2}\right] \\
&= s^2\left[1 + \frac{1}{n} + \frac{(x_p - \bar{x})^2}{\Sigma(x_i - \bar{x})^2}\right]
\end{aligned}
$$

Prediction Interval for an Individual Value of *y*

$$
\begin{aligned}
s_{\text{ind}} &= 13.829\sqrt{1 + \frac{1}{10} + \frac{(10 - 14)^2}{568}} \\
&= 13.829\sqrt{1.1282} \\
&= 14.69
\end{aligned}
$$

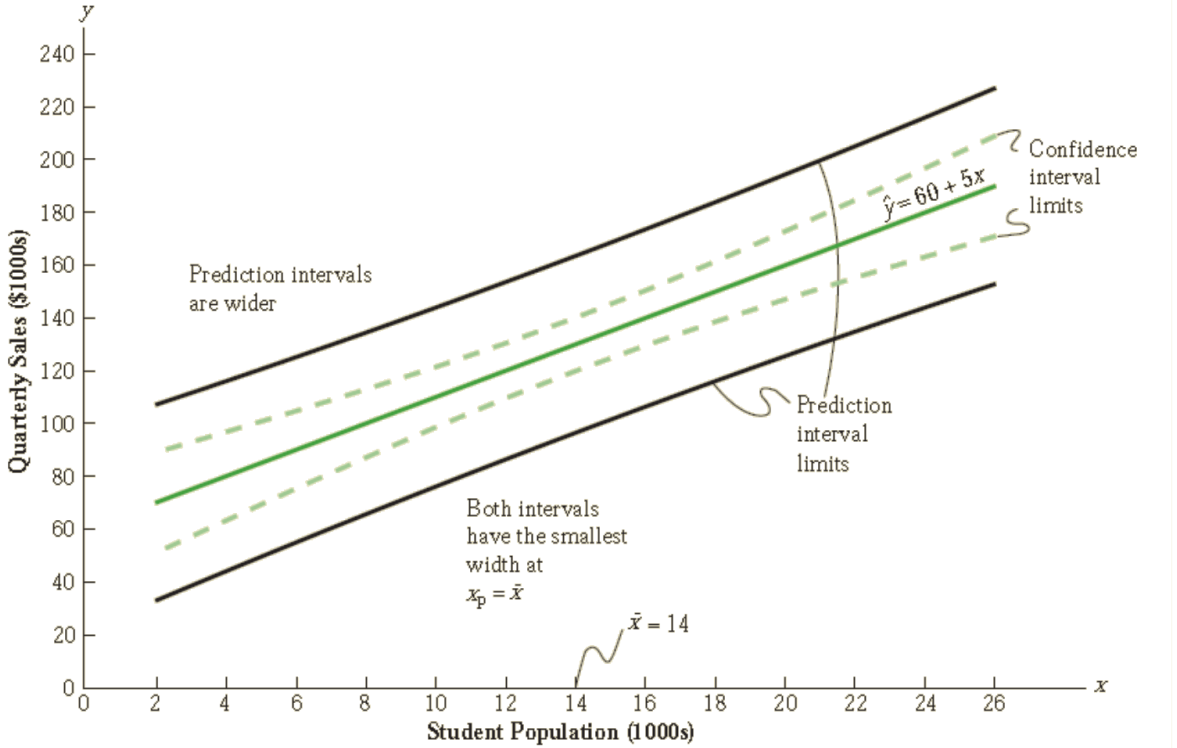
Prediction Interval for an Individual Value of *y*

$$\hat{y}_p \pm t_{\alpha/2} s_{\text{ind}}$$

$$t_{\alpha/2} s_{\text{ind}} = 2.306(14.69) = 33.875,$$

$$110 \pm 33.875$$

Prediction Interval for an Individual Value of *y*



Confidence intervals vs prediction intervals

- Confidence intervals and prediction intervals show the precision of the regression results.
- Narrower intervals provide a higher degree of precision

Python Code for Prediction Interval

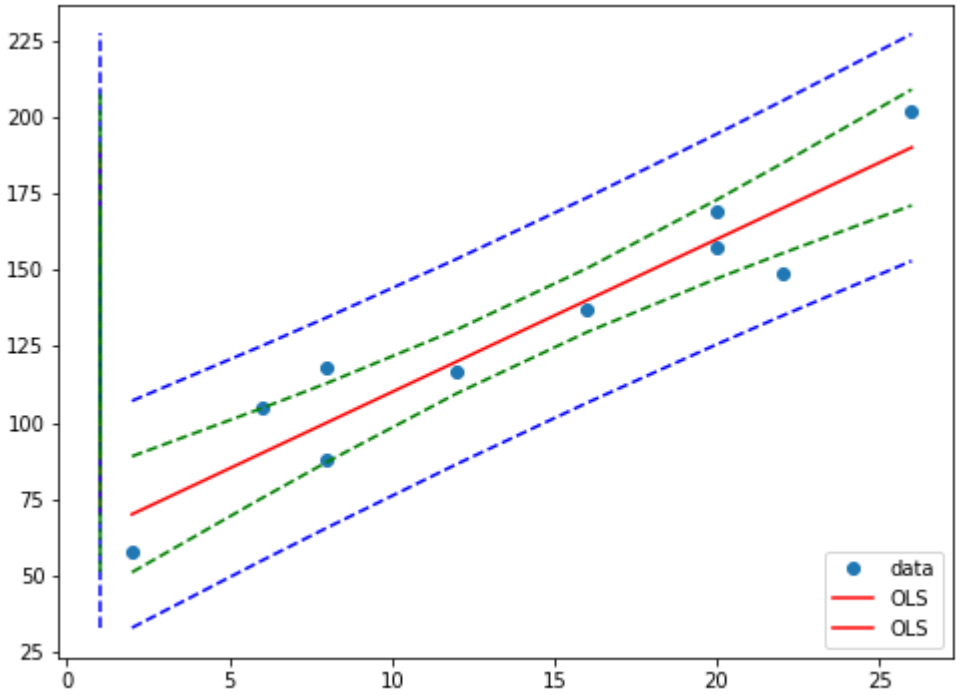
In [43]:
```
from statsmodels.stats.outliers_influence import summary_table

st, data1, ss2 = summary_table(result1, alpha=0.05)
fittedvalues = data1[:,2]
predict_mean_se  = data1[:,3]
predict_mean_ci_low, predict_mean_ci_upp = data1[:,4:6].T
predict_ci_low, predict_ci_upp = data1[:,6:8].T
```

Python Code

```
In [44]: predict_mean_ci_low

Out[44]: array([ 51.03868339,  75.2931351 ,  87.10977127,  87.10977127,
                109.56629808, 129.56629808, 147.10977127, 147.10977127,
                155.2931351 , 171.03868339])

In [45]:  predict_mean_ci_upp

Out[45]: array([ 88.96131661, 104.7068649 , 112.89022873, 112.89022873,
                130.43370192, 150.43370192, 172.89022873, 172.89022873,
                184.7068649 , 208.96131661])

In [46]: predict_ci_low

Out[46]: array([ 32.89834155,  54.8817226 ,  65.60291394,  65.60291394,
                 86.446108  , 106.446108  , 125.60291394, 125.60291394,
                134.8817226 , 152.89834155])

In [47]: predict_ci_upp

Out[47]: array([107.10165845, 125.1182774 , 134.39708606, 134.39708606,
                153.553892  , 173.553892  , 194.39708606, 194.39708606,
                205.1182774 , 227.10165845])
```

Python Code

```
In [48]: X = s.add_constant(x)
         fig, ax = plt.subplots(figsize=(8,6))
         ax.plot(x, y, 'o', label="data")
         ax.plot(X, fittedvalues, 'r-', label='OLS')
         ax.plot(X, predict_ci_low, 'b--')
         ax.plot(X, predict_ci_upp, 'b--')
         ax.plot(X, predict_mean_ci_low, 'g--')
         ax.plot(X, predict_mean_ci_upp, 'g--')
         ax.legend(loc='best');
         plt.show()
```



Thank You