Regression (迴歸)

Dr. Tun-Wen Pai

1) Linear Regression
2) Logistic Regression
3) Log of Odds
4) Examples for Linear/Logistic Regression

April 12, 2023

Regression Introduction

- A **statistical technique** to model the predictive relationship between **several independent variables** and **one dependent variable**.
- Objective is to find the **best-fitting curve** for a dependent variable in a multidimensional space, with each independent variable being a dimension.(Straight lines or nonlinear curves)
- The quality of fit of the curve to the data can be measured by a coefficient of correlation. (**Square root of the amount of variance** explained by the curve).
- Key steps for regression:
  - List all available variables
  - Establish a Dependent Variable (DV) of interest.
  - Examine visual (if possible) relationships between variables of interest.
  - Find a way to predict DV using the other variables.

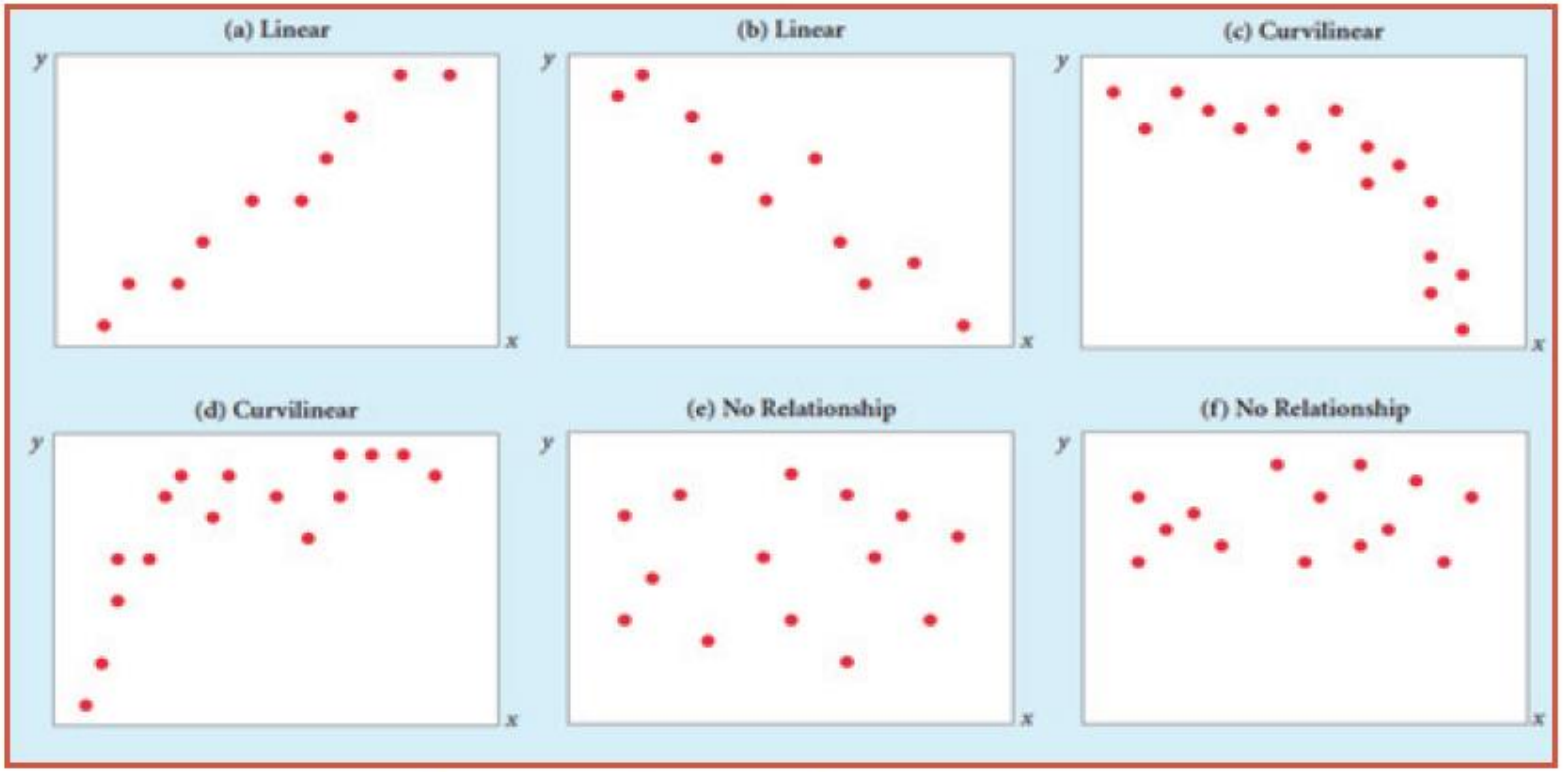
Correlations and Relationships

- Find statistical relationships : categorizing variables that have a relationship with one another, and categorizing variables that are distinct and unrelated to other variables. (Observing significant positive relationships and significant negative differences)
- **Correlation coefficient**

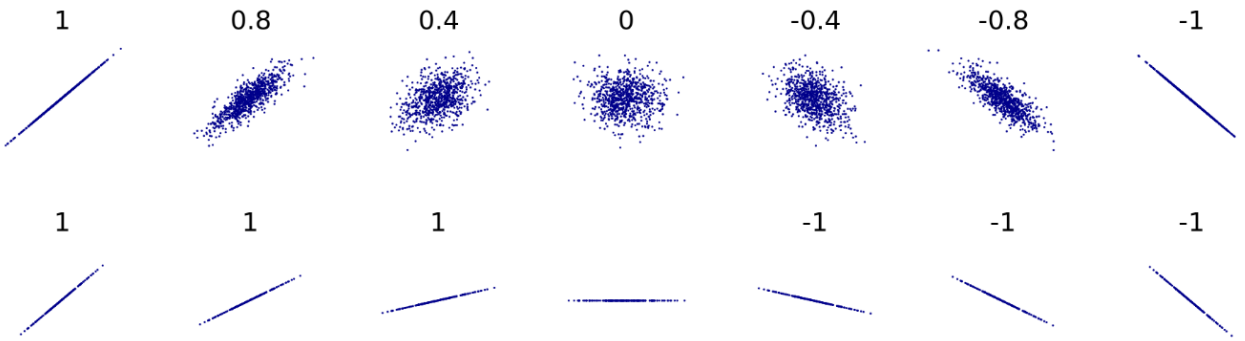
Two variable x and y, correlation coefficient (r) is defined as

$$r = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}}$$

Scatter plots for two variables



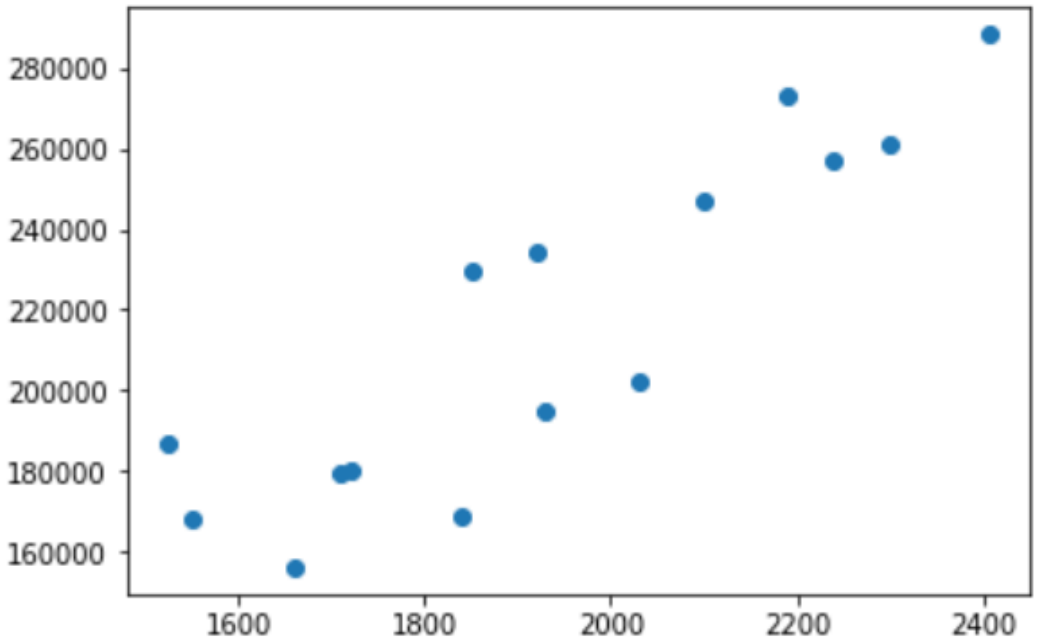
▲圖 7.1：顯示兩變數間各種關係類型的散佈圖（來源：Groebner 等人，2013）



Regression Exercise

房價	面積（平方呎）
$229,500	1850
$273,300	2190
$247,000	2100
$195,100	1930
$261,000	2300
$179,700	1710
$168,500	1550
$234,400	1920
$168,800	1840
$180,400	1720
$156,200	1660
$288,350	2405
$186,750	1525
$202,100	2030
$256,800	2240

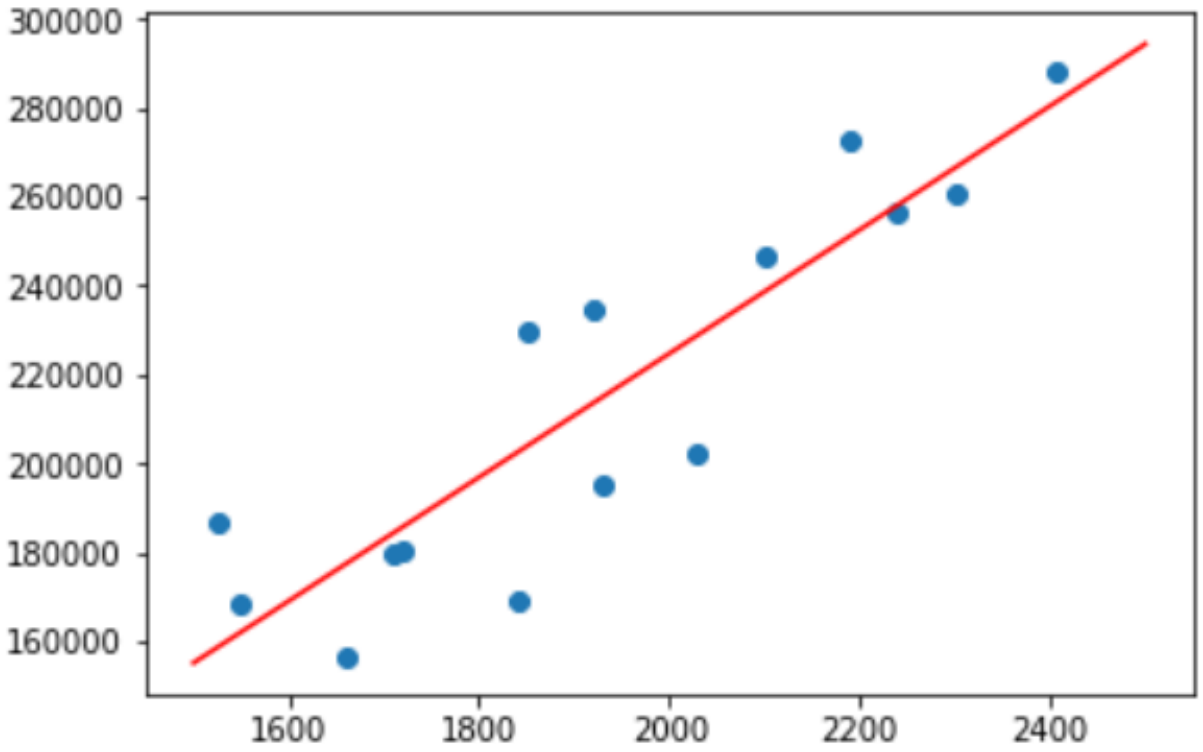
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
x =np.array( [1850, 2190, 2100, 1930, 2300, 1710, 1550, 1920, 1840, 1720, 1660, 2405, 15
y =np.array([229500, 273000, 247000, 195100, 261000, 179700, 168500, 234400, 168800, 180
plt.scatter(x, y);
```



```
from sklearn.linear_model import LinearRegression
model = LinearRegression(fit_intercept=True)

model.fit(x[:, np.newaxis], y)

xfit = np.linspace(1500, 2500, 50)
yfit = model.predict(xfit[:, np.newaxis])
fig, ax=plt.subplots()
ax.scatter(x, y)
ax.plot(xfit, yfit, 'r');
```



```
print("Model slope:    ", model.coef_[0])
print("Model intercept:", model.intercept_)
```

```
Model slope:     139.40748029627565
Model intercept: -54068.98027887367
```

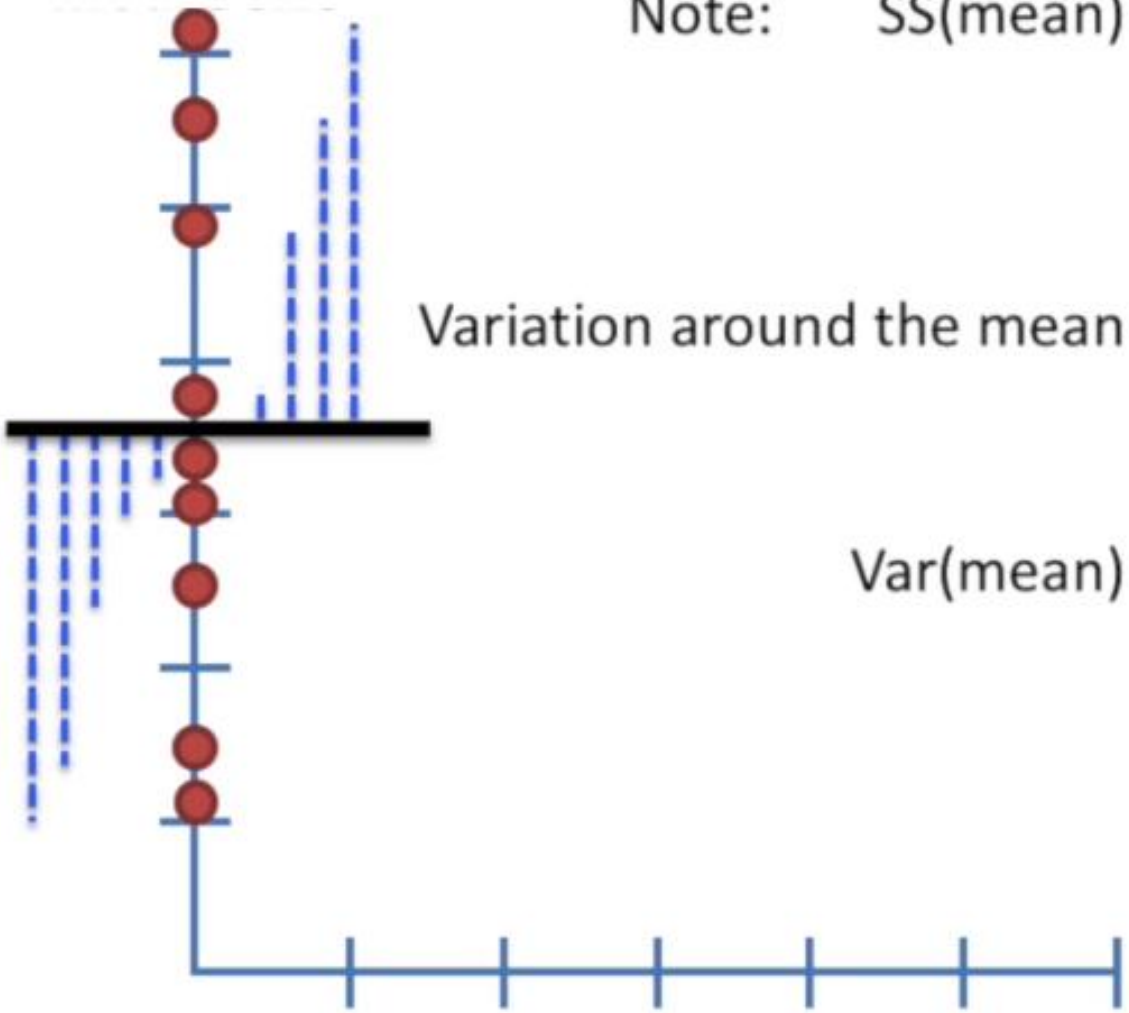
```
model.score(x[:, np.newaxis],y)
```

```
0.794345729393041
```

- It shows the coefficient of correlation between the observed outcomes and the observed predictor values is 0.891 (r), the measure of total variance explained by the equation, is 0.794, or 79%.

Regression Statistics	
r	**0.891**
r^2	**0.794**
	Coefficients
Intercept	**-54191**
Size (sqft)	**139.48**

House Price ($) = 139.48 * Size(sqft) – 54191



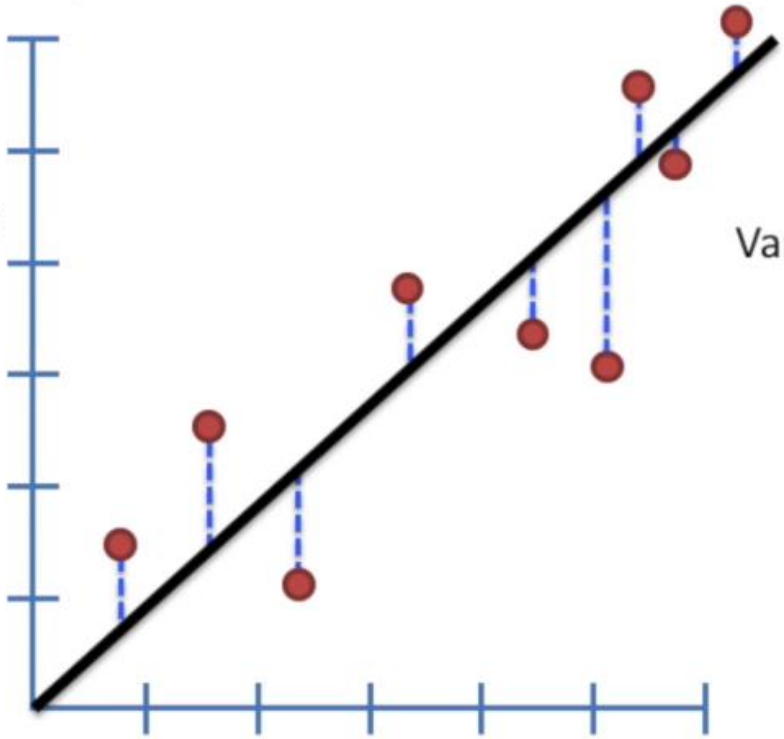
Note: $SS(\text{mean}) = (\text{data} - \text{mean})^2$

$$\text{Variation around the mean} = \frac{(\text{data} - \text{mean})^2}{n}$$

$$\text{Var(mean)} = \frac{SS(\text{mean})}{n}$$

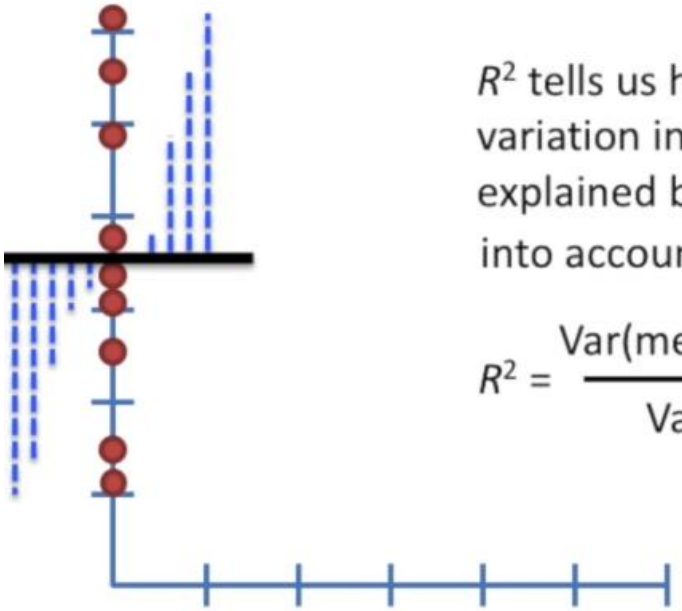
We'll call this **SS(fit)**, for the sum of squares around the least-squares fit.

$SS(fit) = (data - line)^2$



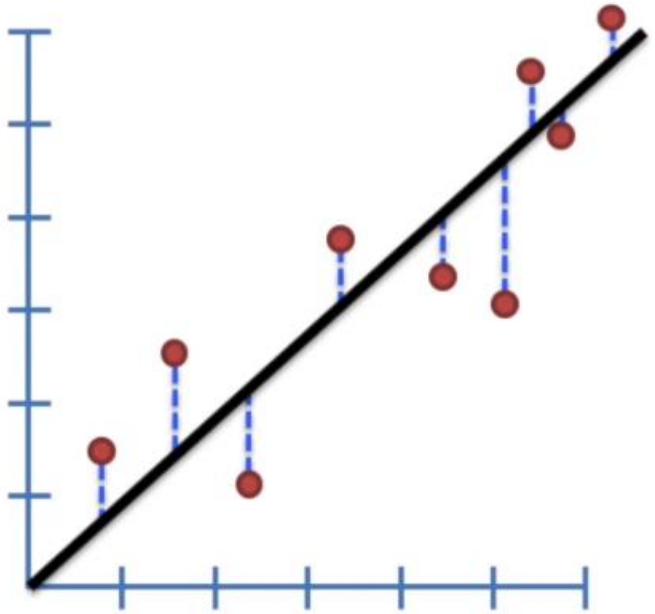
$$Var(fit) = \frac{SS(fit)}{n}$$

In general: Variance(something) = $\frac{\text{Sums of squares}}{\text{The number of those things}}$ = Average sum of squares.



R^2 tells us how much of the variation in **house price** can be explained by taking **house area** into account.

$$R^2 = \frac{\text{Var(mean)} - \text{Var(fit)}}{\text{Var(mean)}}$$



Var(fit) = 0 --> R^2 = 1

The house data now looks like this:

House Price	Size (sqft)	#Rooms
$229,500	1850	4
$273,300	2190	5
$247,000	2100	4
$195,100	1930	3
$261,000	2300	4
$179,700	1710	2
$168,500	1550	2
$234,400	1920	4
$168,800	1840	2
$180,400	1720	2
$156,200	1660	2
$288,350	2405	5
$186,750	1525	3
$202,100	2030	2

While it is possible to make a 3-dimensional scatter plot, one can alternatively examine the correlation matrix among the variables.

	House Price	*Size (sqft)*	*#Rooms*
House Price	1		
Size (sqft)	0.891	1	
Rooms	0.944	0.748	1

Running a regression model between these three variables produces the following output (truncated).

Regression Statistics	
r	**0.984**
r^2	**0.968**

	Coefficients
Intercept	**12923**
Size(sqft)	**65.60**
Rooms	**23613**

House Price ($) = 65.6 * Size (sqft) + 23613 * Rooms + 12924

This equation shows a 97% goodness of fit with the data, which is very good for business and economic data.

This predictive equation should be used for future transactions. Given a situation as below, it will be possible to predict the price of the house with 2000 sq ft and 3 rooms.

House Price	Size (sqft)	#Rooms
??	2000	3

House Price ($) = 65.6 * 2000 (sqft) + 23613 * 3 + 12924 = $214,963

In [54]:
```
from sklearn.linear_model import LinearRegression
model_new = LinearRegression(fit_intercept=True)

model_new.fit(x_new, y)
```

Out[54]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [57]:
```
print("Model slope:    ", model_new.coef_[0],model_new.coef_[1], )
print("Model intercept:", model_new.intercept_)
```

```
Model slope:     65.64756818669008 23576.392133514648
Model intercept: 12941.541814859025
```

In [81]:
```
y_pred = model_new.predict([[2000,3]])
y_pred
```

Out[81]:
```
array([214965.85458878])
```

In [78]:
```
y_pred = model_new.predict(x_new)
y_pred
```

Out[78]:
```
array([228695.11149429, 274591.67681128, 245107.00354097, 210370.52481571,
       258236.5171783 , 172351.66768113, 161848.05677126, 233290.44126736,
       180885.8515454 , 173008.143363  , 169069.28927179, 288705.90397142,
       183783.25970011, 193358.88950087, 254297.6630871 ])
```

In [71]:
```
y
```

Out[71]:
```
array([229500, 273000, 247000, 195100, 261000, 179700, 168500, 234400,
       168800, 180400, 156200, 288350, 186750, 202100, 256800])
```

In [76]:
```
model_new.score(x_new,y)
```

Out[76]:
```
0.9687691294064069
```

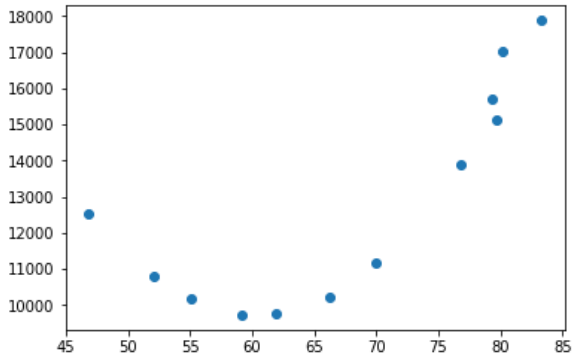
In [79]:
```
model_new.score(x_new, y_pred)
```

Out[79]:
```
1.0
```

Non-linear regression exercise

KWatts	Temp (F)
12530	46.8
10800	52.1
10180	55.1
9730	59.2
9750	61.9
10230	66.2
11160	69.9
13910	76.8
15690	79.3
15110	79.7
17020	80.2
17880	83.3

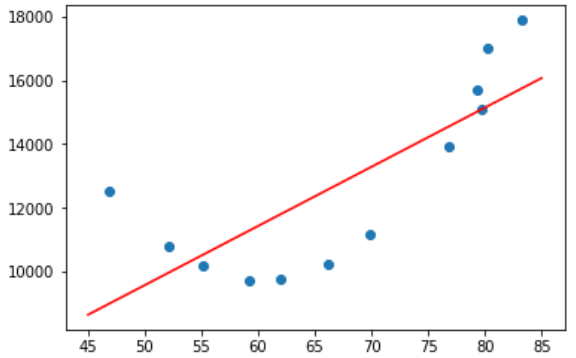
```
KWatts=np.array([12530, 10800, 10180, 9730, 9750, 10230, 11160, 13910, 15690, 15110, 17020, 17880])
Temp=np.array([46.8, 52.1, 55.1, 59.2, 61.9, 66.2, 69.9, 76.8, 79.3, 79.7, 80.2, 83.3])
plt.scatter(Temp, KWatts);
```




```
from sklearn.linear_model import LinearRegression
model_2 = LinearRegression(fit_intercept=True)

model_2.fit(Temp[:, np.newaxis], KWatts)

x_test = np.linspace(45, 85, 30)
y_pred = model_2.predict(x_test[:, np.newaxis])
fig, ax=plt.subplots()
ax.scatter(Temp, KWatts)
ax.plot(x_test, y_pred, 'r');
```



Thus, only 60% of the variance is explained.

Regression Statistics	
r	0.992
r^2	0.984
	Coefficients
Intercept	**67245**
Temp (F)	**-1911**
Temp-sq	**15.87**

```
x_3 = pd.DataFrame({'Temp':Temp ,'Temp_2':np.square(Temp)},
                   columns =['Temp','Temp_square'])
x_3['Temp_square']=np.square(Temp)
```

```
from sklearn.linear_model import LinearRegression
model_3 = LinearRegression(fit_intercept=True)

model_3.fit(x_3, KWatts)
print("Model slope:    ", model_3.coef_[0],model_3.coef_[1], )
print("Model intercept:", model_3.intercept_)
```

```
Model slope:     -1911.0388414791091 15.870041855256996
Model intercept: 67245.1685313067
```

```
y_pred = model_3.predict(x_3)
y_pred
```

```
array([12567.75122314, 10757.84520257, 10128.53413879,  9730.45260335,
        9759.67531677, 10283.90345354, 11204.73671712, 14082.70117806,
       15498.38790838, 15743.30703378, 16056.59745937, 18176.10776512])
```

```
model_3.score(x_3,KWatts)
```

```
0.9846710121739651
```

Energy Consumption (Kwatts) = 15.87 * Temp^2 -1911 * Temp + 67245

This equation shows a 98.5% fit which is very good for business and economic contexts. Now one can predict the Kwatts value for when the temperature is 72-degrees.

Energy consumption = (15.87 * 72*72) - (1911 * 72) + 67245 = 11923 Kwatts

Derivation of linear regression equations

The mathematical problem is straightforward:

given a set of n points (X_i, Y_i) on a scatterplot,
find the best-fit line, $\hat{Y}_i = a + bX_i$
such that the sum of squared errors in Y, $\sum \left(Y_i - \hat{Y}_i\right)^2$ is minimized

The derivation proceeds as follows: for convenience, name the sum of squares "Q",

$$Q = \sum_{i=1}^{n} \left(Y_i - \hat{Y}\right)^2 = \sum_{i=1}^{n} \left(Y_i - a - bX_i\right)^2 \qquad (1)$$

Then, Q will be minimized at the values of a and b for which $\partial Q / \partial a = 0$ and $\partial Q / \partial b = 0$.

The first of these conditions is,

$$\frac{\partial Q}{\partial a} = \sum_{i=1}^{n} -2\left(Y_i - a - bX_i\right) = 2\left(na + b\sum_{i=1}^{n} X_i - \sum_{i=1}^{n} Y_i\right) = 0 \qquad (2)$$

which, if we divide through by 2 and solve for *a*, becomes simply,

$$a = \overline{Y} - b\overline{X} \qquad (3)$$

which says that the constant *a* (the y-intercept) is set such that the line must go through the mean of x and y.

The second condition for minimizing Q is,

$$\frac{\partial Q}{\partial b} = \sum_{i=1}^{n} -2X_i\left(Y_i - a - bX_i\right) = \sum_{i=1}^{n} -2\left(X_iY_i - aX_i - bX_i^2\right) = 0 \qquad (4)$$

If we substitute the expression for *a* from (3) into (4), then we get,

$$\sum_{i=1}^{n}\left(X_iY_i - X_i\bar{Y} + bX_i\bar{X} - bX_i^2\right) = 0 \qquad (5)$$

We can separate this into two sums,

$$\sum_{i=1}^{n}\left(X_iY_i - X_i\bar{Y}\right) - b\sum_{i=1}^{n}\left(X_i^2 - X_i\bar{X}\right) = 0 \qquad (6)$$

which becomes directly,

$$b = \frac{\sum_{i=1}^{n}\left(X_iY_i - X_i\bar{Y}\right)}{\sum_{i=1}^{n}\left(X_i^2 - X_i\bar{X}\right)} = \frac{\sum_{i=1}^{n}\left(X_iY_i\right) - n\bar{X}\,\bar{Y}}{\sum_{i=1}^{n}\left(X_i^2\right) - n\bar{X}^2} \qquad (7)$$

$$b = \frac{\sum_{i=1}^{n}\left(X_i Y_i - X_i \overline{Y}\right)}{\sum_{i=1}^{n}\left(X_i^2 - X_i \overline{X}\right)} = \frac{\sum_{i=1}^{n}\left(X_i Y_i\right) - n\overline{X}\,\overline{Y}}{\sum_{i=1}^{n}\left(X_i^2\right) - n\overline{X}^2} \qquad (7)$$

We can translate (7) into a more intuitively obvious form, by noting that

$$\sum_{i=1}^{n}\left(\overline{X}^2 - X_i \overline{X}\right) = 0 \qquad \text{and} \qquad \sum_{i=1}^{n}\left(\overline{X}\,\overline{Y} - Y_i \overline{X}\right) = 0 \qquad (8)$$

so that b can be rewritten as the ratio of Cov(x,y) to Var(x):

$$b = \frac{\sum_{i=1}^{n}\left(X_i Y_i - X_i \overline{Y}\right) + \sum_{i=1}^{n}\left(\overline{X}\,\overline{Y} - Y_i \overline{X}\right)}{\sum_{i=1}^{n}\left(X_i^2 - X_i \overline{X}\right) + \sum_{i=1}^{n}\left(\overline{X}^2 - X_i \overline{X}\right)} = \frac{\frac{1}{n}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)\left(Y_i - \overline{Y}\right)}{\frac{1}{n}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2} = \frac{Cov(X,Y)}{Var(X)} \qquad (9)$$

The quantities that result from regression analyses can be written in many different forms that are mathematically equivalent but superficially distinct. All of the following forms of the regression slope b are mathematically equivalent:

$$b = \frac{Cov(X,Y)}{Var(X)} \text{ or } \frac{\sum xy}{\sum x^2} \text{ or } \frac{\sum_{i=1}^{n}\left(X_i Y_i\right) - \frac{\sum_{i=1}^{n} X_i \sum_{i=1}^{n} Y_i}{n}}{\sum_{i=1}^{n}\left(X_i\right)^2 - \frac{\left(\sum_{i=1}^{n} X_i\right)^2}{n}} \text{ or } \frac{\sum_{i=1}^{n}\left(X_i Y_i\right) - n\overline{X}\,\overline{Y}}{\sum_{i=1}^{n}\left(X_i^2\right) - n\overline{X}^2} \text{ or } \frac{\frac{1}{n}\sum_{i=1}^{n}\left(X_i Y_i\right) - \overline{X}\,\overline{Y}}{\frac{1}{n}\sum_{i=1}^{n}\left(X_i^2\right) - \overline{X}^2} \text{ or } \frac{\overline{(XY)} - \overline{X}\,\overline{Y}}{\overline{\left(X_i^2\right)} - \overline{X}^2} \qquad (10)$$

A common notational shorthand is to write the "sum of squares of X" (that is, the sum of squared deviations of the X's from their mean), the "sum of squares of Y", and the "sum of XY cross products" as,

$$\sum x^2 = SS_x = (n-1)Var(X) = \sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2 = \sum_{i=1}^{n}\left(X_i^2\right) - n\overline{X}^2 \qquad (11)$$

$$\sum y^2 = SS_y = (n-1)Var(Y) = \sum_{i=1}^{n}\left(Y_i - \overline{Y}\right)^2 = \sum_{i=1}^{n}\left(Y_i^2\right) - n\overline{Y}^2 \qquad (12)$$

$$\sum xy = S_{xy} = (n-1)Cov(X,Y) = \sum_{i=1}^{n}\left(X_i - \overline{X}\right)\left(Y_i - \overline{Y}\right) = \sum_{i=1}^{n}\left(X_iY_i\right) - n\overline{X}\,\overline{Y} \qquad (13)$$

It is important to recognize that Σx^2, Σy^2, and Σxy, as used in Zar and in equations (10)-(13), are not summations; instead, they are *symbols* for the sums of squares and cross products. Note also that S and SS in (11)-(13) are uppercase S's rather than standard deviations.

Besides the regression slope *b* and intercept *a*, the third parameter of fundamental importance is the correlation coefficient *r* or the coefficient of determination r^2. r^2 is the ratio between the variance in Y that is "explained" by the regression (or, equivalently, the variance in $\hat{Y}$), and the total variance in Y. Like *b*, r^2 can be calculated many different ways:

$$r^2 = \frac{Var(\hat{Y})}{Var(Y)} = \frac{b^2Var(X)}{Var(Y)} = \frac{(Cov(x,y))^2}{Var(X)Var(Y)} = \frac{Var(Y) - Var(Y-\hat{Y})}{Var(Y)} = \frac{S_{xy}^2}{SS_xSS_y} \qquad (14)$$

Logistic Regression

- Why can't we use Linear Regression to model binary responses?

→ Linear regression is the type of regression we use for a continuous, normally distributed response (Y) variable

- The response (Y) is NOT normally distributed
- The variability of Y is NOT constant
  - Variance of Y depends on the expected value of Y
  - For a Y~Binomial(n,p), $f(k;n,p) = \Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$

  we have Var(Y)=np(1-p) which depends on the expected response, E(Y)=np
- The model must produce predicted/fitted probabilities that are between 0 and 1
  - Linear models produce fitted responses that vary from -∞ to ∞

Logistic regression

- Dependent variable is **categorical variables** (nominal or ordinal dependent variables)
- For example: voting, morbidity or mortality, and participation data is not continuous or distributed normally.
- Binary logistic regression is a type of regression analysis where the dependent variable is a dummy variable: coded 0 (did not vote) or 1(did vote)

The Logistic Regression Model

Logistic Regression ("logit" regression model)

$$\ln[p/(1-p)] = \alpha + \beta X$$

- p is the probability that the event Y occurs (success), p(Y=1)
- 1-p is the probability that the event Y does not occur(fail), p(Y=0)
- p/(1-p) is the "odds"
- ln[p/(1-p)] is the log odds, or "logit"

勝算(Odds)：$\frac{p}{1-p} = e^{f(x)}$，即事件成功機率與失敗機率的比值

	事件成功	事件失敗	總和
實驗組	4 (a)	16 (b)	20 (a+b)
對照組	1 (c)	19 (d)	20 (c+d)
	5 (a+c)	35 (b+d)	40 (a+b+c+d)

實驗組的勝算(Experimental event odds) = a/b = 4/16 = 0.25

控制組的勝算(Control event odds) = c/d = 1/19=0.053

勝算比(odds ratio) = (a/b) / (c/d) = ad/bc = 4.72

More:

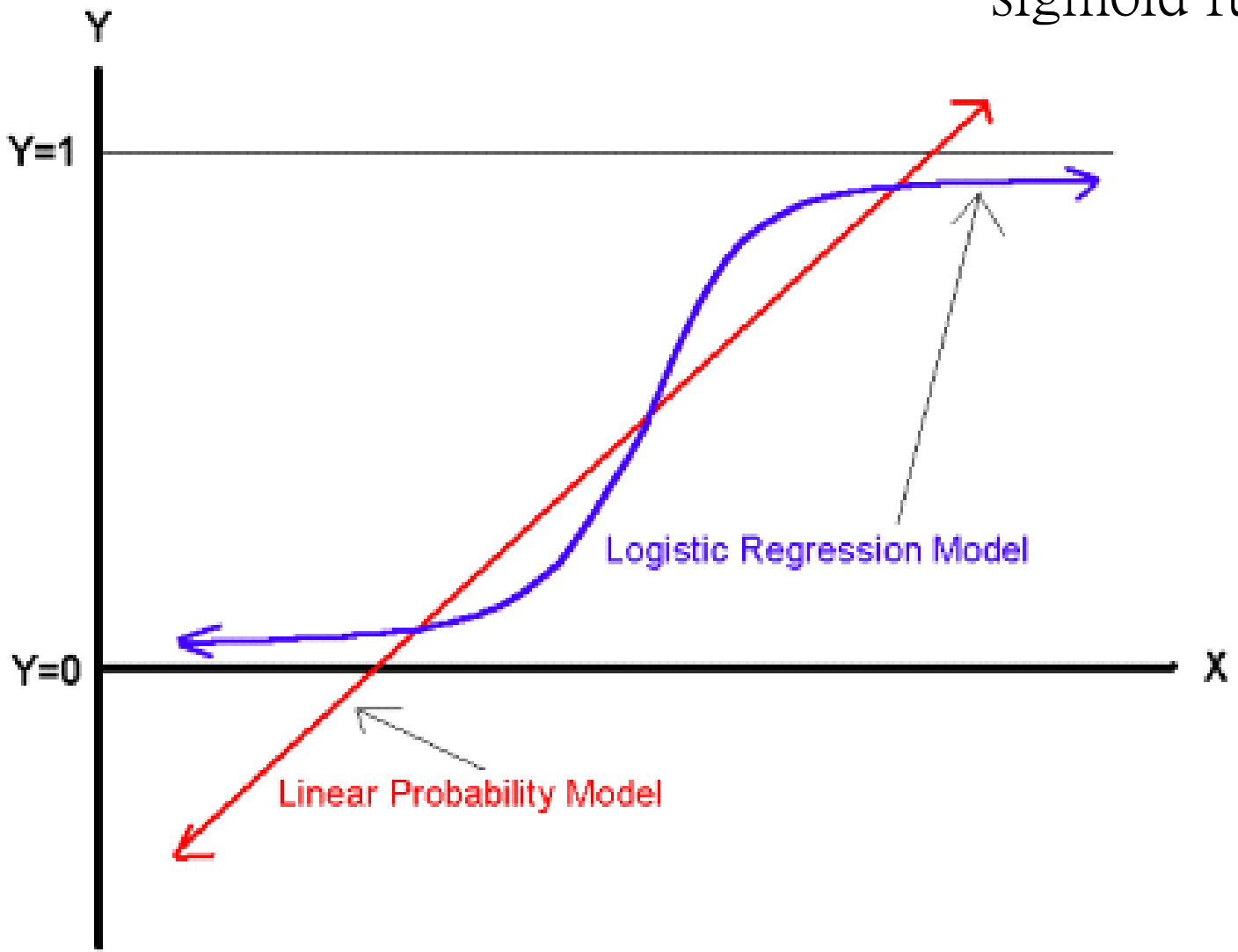
- The logistic distribution constrains the estimated probabilities to lie between 0 and 1.
- The estimated probability is:

$$p = p(Y=1) = \exp(\alpha + \beta X)/[1 + \exp(\alpha + \beta X)]$$
$$= 1/[1 + \exp(-\alpha - \beta X)]$$
$$p(Y=0) = 1-p = 1/[1 + \exp(\alpha + \beta X)]$$

- if you let $\alpha + \beta X = 0$, then p = .50
- as $\alpha + \beta X$ gets really big, p approaches 1
- as $\alpha + \beta X$ gets really small, p approaches 0

Comparing the LP and Logit Models

sigmoid function



- An interpretation of the logit coefficient which is usually more intuitive is the "odds of the event"
- Since:

$$[p/(1-p)] = \exp(\alpha + \beta X)$$

- $\ln([p/(1-p)] = \ln(\exp(\alpha + \beta X)) = \alpha + \beta X$

β increment implies the change of odds of event

→ using $\exp(\beta)$ instead of β to describe the effect of the independent variable on the "odds lf event"

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
from sklearn.datasets import load_iris
```

```
from sklearn.linear_model import LogisticRegression
```

```
X, y = load_iris(return_X_y=True)
```

```
clf = LogisticRegression(random_state=0, solver='sag',multi_class='multinomial')
```

```
X[:5]
```

```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2]])
```

```
clf=clf.fit(X,y)
```

```
C:\Users\te\Anaconda3\lib\site-packages\sklearn\linear_model\sag.py:334: ConvergenceWarning: The max_iter was reached which mea
ns the coef_ did not converge
  "the coef_ did not converge", ConvergenceWarning)
```

```
clf.predict([[4.7, 3.2, 1.3, 0.2], [6.3, 3.3, 6. , 2.5]])
```

```
array([0, 2])
```

```
clf.predict(X[:2, :])
```

```
array([0, 0])
```

```
clf.predict_proba(X[:2, :])
```

```
array([[9.82977664e-01, 1.70222296e-02, 1.05889447e-07],
       [9.54909075e-01, 4.50903038e-02, 6.20765476e-07]])
```

```
clf.classes_
```

```
array([0, 1, 2])
```

```
clf.coef_
```

```
array([[ 0.61882374,  1.61956745, -2.36973754, -1.10174388],
       [ 0.37397659, -0.28872847, -0.03389596, -0.95622952],
       [-0.99280033, -1.33083898,  2.4036335 ,  2.0579734 ]])
```

```
clf.intercept_
```

```
array([ 1.41341676,  1.98582553, -3.39924229])
```

```
clf.score(X, y)
```

```
0.9866666666666667
```

提醒:
下週三(4/25/2023)小考: 範圍至Regression Model
4/26/2023(三)期中考: 範圍至Regression Model

- 作業: 使用以下資料建立一份線性迴歸模型，從Test1 分數中預測Test2。
- 計算回歸模型的R^2 (R-squared, Coefficient of Determination)
- 預測如果學生在Test1取得的分數為60分，Test2會獲得的分數為何?

HW Due: 4/28/2023 (五)

Test1	Test2
59	56
52	63
44	55
51	50
42	66
42	48
41	58
45	36
27	13
63	50
54	81
44	56
50	64
47	50