scikit-learn

監督式學習是機器學習中最常用的、也是較為成功的模式

監督式學習模式：

- 模型建構在輸入與輸出配對之訓練資料，然後對於尚未見過的資料進行預測，訓練資料通常需要人工建立

常見的監督式學習演算法：

- 線性模型 (Linear models)
- 決策樹 (Decision trees)、隨機森林 (Random forests)
- 最近鄰居法 (k-Nearest Neighbors)、支援向量機(SupportVectorMachine)
- 類神經網路 (Neural networks) …等

scikit-learn

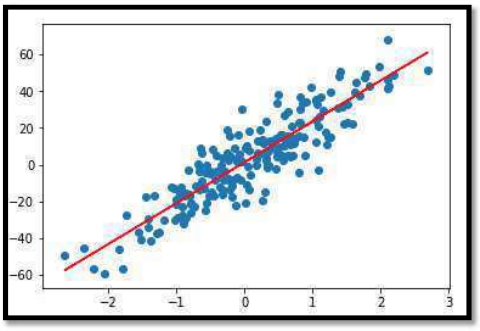
監督式學習分為兩種型態：分類與迴歸

- 迴歸 (Regression)：預測**連續數值** (實數、浮點數)
  - 例如：預測農產品的收成量
- 分類 (Classification)：從已定義的離散標籤中預測資料屬於哪種標籤(**類比數值**)
  - 只有兩種標籤的問題稱為二元分類 (Binary classification)，常用在答案是「是/否」的問題， 例如：信件是否為垃圾郵件？
  - 有多種標籤的問題稱為多(元)類別分類 (Multiclass classification)，例如：鳶尾花屬於三個品種裡的哪個品種？

scikit-learn

線性迴歸(Linear Regression)

- 線性迴歸(Linear Regression)可以說是機器學習入門方法之一，也是常見的統計方法。它的目的在於透過過去資料找出已經存在的關係，並且利用X來預測Y可能的結果。線性迴歸模型指的是X(自變數)及Y(依變數)的線性關係，依自變數多寡可分成**簡單線性迴歸及複(多元)迴歸**。迴歸分析又可依照函數型態區分為**線性及非線性**。
- 舉例來說：簡單線性迴歸模型y=ax+b，a和b稱為迴歸係數(regression coefficients)，為了要找出迴歸係數，常用的方法為**最小平方法(Ordinary Least Squares, OLS)**，找出與各點殘差(Residual)最小的公式。



scikit-learn

線性迴歸(Linear Regression)

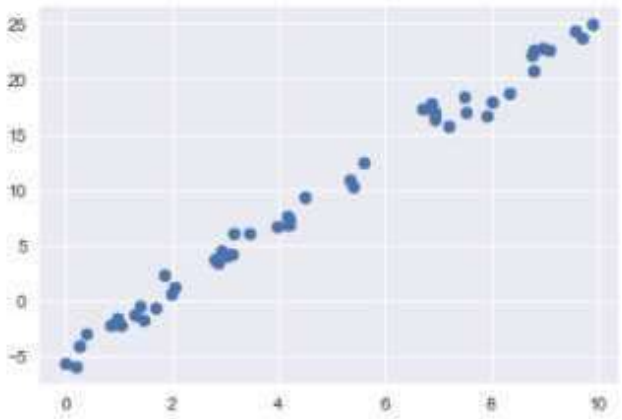
- 線性迴歸將預測 $\hat{y}$ 與真實迴歸目標 y 之間的誤差最小化來學習參數 w 與 b：
  - 預測值：$\hat{y}_i = wx_i + b$
  - 偏差 (Deviation)：$d_i = \hat{y}_i - y_i = (wx_i + b) - y_i$ ← 觀察值到迴歸線(預測值)的垂直距離
  - 誤差函數 (Error function)：$E(\hat{y}, y) = \sum_i (\hat{y}_i - y_i)^2 = \sum_i (wx_i + b - y_i)^2$ ← 平方差
  - 擬合：將E最小化
    - 設定 $\partial E/\partial w = 0$ 與 $\partial E/\partial b = 0$ → 解聯立方程式可求得 w 與 b

scikit-learn

線性迴歸(Linear Regression)

- 考慮到使用的數據，如下所舉例斜率為3，截距為-5。

```
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
import numpy as np

rng = np.random.RandomState(1)
x = 10 * rng.rand(50)
y = 3 * x - 5 + rng.randn(50)
plt.scatter(x, y);
```

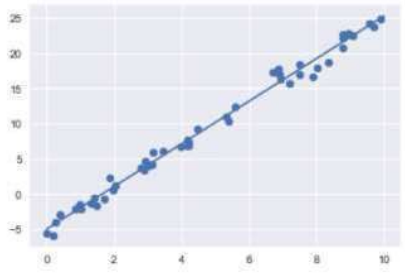


scikit-learn

線性迴歸(Linear Regression)

- 再來，使用SKlearn中的LinearRegression模組來擬合數據，並利用plt.plot()方式建構繪製出最適合的線。

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

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

xfit = np.linspace(0, 10, 1000)
yfit = model.predict(xfit[:, np.newaxis])

plt.scatter(x, y)
plt.plot(xfit, yfit);
```



- 而模型的斜率及截距，分別儲存在model.coef_[0] 和 model.intercept_中。

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

```
Model slope:     3.0272088103606944
Model intercept: -4.998577085553199
```

scikit-learn

建立線性迴歸分析模型：用氣溫來預測冰紅茶的銷售量

- 使用 sklearn.linear_model 的 LinearRegression() 方法。

```
import numpy as np
from sklearn.linear_model import LinearRegression

temperatures = np.array([29, 28, 34, 31, 25, 29, 32, 31, 24, 33, 25, 31, 26, 3
0])
iced_tea_sales = np.array([77, 62, 93, 84, 59, 64, 80, 75, 58, 91, 51, 73, 65,
 84])

lm = LinearRegression()
lm.fit(np.reshape(temperatures, (len(temperatures), 1)), np.reshape(iced_tea_s
ales, (len(iced_tea_sales), 1)))

# 印出係數
print(lm.coef_)

# 印出截距
print(lm.intercept_ )
```

```
[[ 3.73788546]]
[-36.36123348]
```

scikit-learn

利用線性迴歸分析模型預測：用氣溫來預測冰紅茶的銷售量

- 建立線性迴歸模型之後，身為冰紅茶店的老闆，就可以開始量測氣溫，藉此來預測冰紅茶銷量，更精準地掌握原料的管理。
- 使用LinearRegression()的predict()方法。

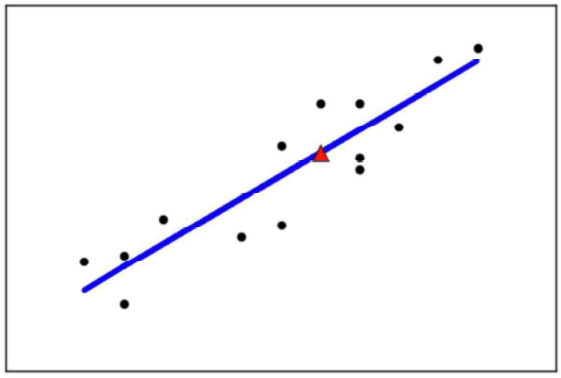
```
import numpy as np
from sklearn.linear_model import LinearRegression

temperatures = np.array([29, 28, 34, 31, 25, 29, 32, 31, 24, 33, 25, 31, 26, 3
0])
iced_tea_sales = np.array([77, 62, 93, 84, 59, 64, 80, 75, 58, 91, 51, 73, 65,
 84])

lm = LinearRegression()
lm.fit(np.reshape(temperatures, (len(temperatures), 1)), np.reshape(iced_tea_s
ales, (len(iced_tea_sales), 1)))

# 新的氣溫
to_be_predicted = np.array([30])
predicted_sales = lm.predict(np.reshape(to_be_predicted, (len(to_be_predicte
d), 1)))

# 預測的冰紅茶銷量
print(predicted_sales)
```

```
[[ 75.7753304]]
```

scikit-learn

線性迴歸視覺化

- 使用matplotlib.pyplot的scatter()與plot()方法。



```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression

temperatures = np.array([29, 28, 34, 31, 25, 29, 32, 31, 24, 33, 25, 31, 26, 3
0])
iced_tea_sales = np.array([77, 62, 93, 84, 59, 64, 80, 75, 58, 91, 51, 73, 65,
 84])

lm = LinearRegression()
lm.fit(np.reshape(temperatures, (len(temperatures), 1)), np.reshape(iced_tea_s
ales, (len(iced_tea_sales), 1)))

# 新的氣溫
to_be_predicted = np.array([30])
predicted_sales = lm.predict(np.reshape(to_be_predicted, (len(to_be_predicte
d), 1)))

# 視覺化
plt.scatter(temperatures, iced_tea_sales, color='black')
plt.plot(temperatures, lm.predict(np.reshape(temperatures, (len(temperatures),
 1))), color='blue', linewidth=3)
plt.plot(to_be_predicted, predicted_sales, color = 'red', marker = '^', marker
size = 10)
plt.xticks(())
plt.yticks(())
plt.show()
```

scikit-learn

線性迴歸模型的績效

- 線性迴歸模型的績效(Performance)有Mean squared error(MSE)與R-squared。

```
import numpy as np
from sklearn.linear_model import LinearRegression

temperatures = np.array([29, 28, 34, 31, 25, 29, 32, 31, 24, 33, 25, 31, 26, 3
0])
iced_tea_sales = np.array([77, 62, 93, 84, 59, 64, 80, 75, 58, 91, 51, 73, 65,
 84])

# 轉換維度
temperatures = np.reshape(temperatures, (len(temperatures), 1))
iced_tea_sales = np.reshape(iced_tea_sales, (len(iced_tea_sales), 1))

lm = LinearRegression()
lm.fit(temperatures, iced_tea_sales)

# 模型績效
mse = np.mean((lm.predict(temperatures) - iced_tea_sales) ** 2)
r_squared = lm.score(temperatures, iced_tea_sales)

# 印出模型績效
print(mse)
print(r_squared)
```

```
27.9348646948
0.822509288117
```

scikit-learn

線性迴歸模型的績效

- Mean Absolute Error(MAE) 代表平均誤差，公式為所有**實際值及預測值相減的絕對值平均**。
  - metrics.mean_absolute_error(y_test,predictions)
- Mean Squared Error(MSE) 比起MAE可以拉開誤差差距，算是常用的指標，公式為所有**實際值及預測值相減的平方的平均**。
  - metrics.mean_squared_error(y_test,predictions)
- Root Mean Squared Error(RMSE) 代表MSE的**平方**根。比起MSE更為常用，因為更容易解釋y。
  - np.sqrt(metrics.mean_squared_error(y_test,predictions))

Mean squared error	$\mathrm{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	$\mathrm{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$