In [2]:
```
import sklearn.datasets
import pandas as pd
import numpy as np

from sklearn.metrics import mean_squared_error

from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import RidgeCV

from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
```

In [15]:
```
X,y = sklearn.datasets.load_diabetes(return_X_y=True, as_frame=True)
print('原始資料\n{X}\n\n資料結構：\n\n{y}\n\n'.format(X=X.head(5), y=y.head(5)))
X_without_bp = X.drop('bp', axis=1)
print('移除bp資料\n{X}\n\n資料結構：{y}\n\n'.format(X=X_without_bp.head(5), y=y.head(5)))
```

In [15]:
```
X,y = sklearn.datasets.load_diabetes(return_X_y=True, as_frame=True)
print('原始資料\n{X}\n\n資料結構：\n\n{y}\n\n'.format(X=X.head(5), y=y.head(5)))
X_without_bp = X.drop('bp', axis=1)
print('移除bp資料\n{X}\n\n資料結構：{y}\n\n'.format(X=X_without_bp.head(5), y=y.head(5)))
```

```
原始資料
        age       sex       bmi        bp        s1        s2        s3  \
0  0.038076  0.050680  0.061696  0.021872 -0.044223 -0.034821 -0.043401
1 -0.001882 -0.044642 -0.051474 -0.026328 -0.008449 -0.019163  0.074412
2  0.085299  0.050680  0.044451 -0.005671 -0.045599 -0.034194 -0.032356
3 -0.089063 -0.044642 -0.011595 -0.036656  0.012191  0.024991 -0.036038
4  0.005383 -0.044642 -0.036385  0.021872  0.003935  0.015596  0.008142

         s4        s5        s6
0 -0.002592  0.019908 -0.017646
1 -0.039493 -0.068330 -0.092204
2 -0.002592  0.002864 -0.025930
3  0.034309  0.022692 -0.009362
4 -0.002592 -0.031991 -0.046641

資料結構：

0    151.0
1     75.0
2    141.0
3    206.0
4    135.0
Name: target, dtype: float64
```

資料結構：

```
0    151.0
1     75.0
2    141.0
3    206.0
4    135.0
Name: target, dtype: float64
```

移除bp資料

```
        age       sex       bmi        s1        s2        s3        s4  \
0  0.038076  0.050680  0.061696 -0.044223 -0.034821 -0.043401 -0.002592
1 -0.001882 -0.044642 -0.051474 -0.008449 -0.019163  0.074412 -0.039493
2  0.085299  0.050680  0.044451 -0.045599 -0.034194 -0.032356 -0.002592
3 -0.089063 -0.044642 -0.011595  0.012191  0.024991 -0.036038  0.034309
4  0.005383 -0.044642 -0.036385  0.003935  0.015596  0.008142 -0.002592

         s5        s6
0  0.019908 -0.017646
1 -0.068330 -0.092204
2  0.002864 -0.025930
3  0.022692 -0.009362
4 -0.031991 -0.046641
```

資料結構：0   151.0

```
1     75.0
2    141.0
3    206.0
4    135.0
Name: target, dtype: float64
```

In [40]:
```
kf = KFold(n_splits=5, shuffle=True)
MSE = 0
for train_index, test_index in kf.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y[train_index], y[test_index]
    ridge_reg = Ridge(alpha=0.1).fit(X_train, y_train)
    y_pred = ridge_reg.predict(X_test)
    MSE += mean_squared_error(y_test, y_pred)

print('MSE=', MSE/5)

print('coefficent={coef}\n'.format(coef=ridge_reg.coef_))
print('intercept={inter}\n'.format(inter=ridge_reg.intercept_))
print('model:Y={inter}'.format(inter=ridge_reg.intercept_), end="")
for i, name in enumerate(X.columns):
    print('+{coef}*X[{name}]\n'.format(coef=ridge_reg.coef_[i], name=name), end="")
```

```
MSE= 3021.9578946914503
coefficent=[   3.65039374 -240.57882194  431.54111639  336.03743754  -75.41798617
  -86.61127621 -179.0614262   139.82920273  449.08637648   75.36243502]

intercept=150.7871623652672

model:Y=150.7871623652672+3.6503937367702717*X[age]
+-240.57882193845265*X[sex]
+431.54111639310844*X[bmi]
+336.03743754034974*X[bp]
+-75.41798617220809*X[s1]
+-86.61127620647864*X[s2]
+-179.0614262029827*X[s3]
+139.8292027327749*X[s4]
+449.08637648233724*X[s5]
+75.36243502130375*X[s6]
```

In [42]:
```
kf = KFold(n_splits=5, shuffle=True)
MSE = 0
for train_index, test_index in kf.split(X_without_bp):
    X_train, X_test = X_without_bp.iloc[train_index], X_without_bp.iloc[test_index]
    y_train, y_test = y[train_index], y[test_index]
    ridge_reg = Ridge(alpha=0.1).fit(X_train, y_train)
    y_pred = ridge_reg.predict(X_test)
    MSE += mean_squared_error(y_test, y_pred)

print('MSE=', MSE/5)

print('coefficent={coef}\n'.format(coef=ridge_reg.coef_))
print('intercept={inter}\n'.format(inter=ridge_reg.intercept_))
print('model:Y={inter}'.format(inter=ridge_reg.intercept_), end="")
for i, name in enumerate(X_without_bp.columns):
    print('+{coef}*X[{name}]\n'.format(coef=ridge_reg.coef_[i], name=name), end="")
```

```
MSE= 3200.657396864584
coefficent=[  51.38241582 -153.02618192  515.22577229  -42.42802892  -97.07399058
 -163.8523462    93.64506375  505.86475751  205.05717667]

intercept=150.6941433781982

model:Y=150.6941433781982+51.38241582454106*X[age]
+-153.02618191873853*X[sex]
+515.2257722868852*X[bmi]
+-42.42802892084366*X[s1]
+-97.0739905761172*X[s2]
+-163.85234619580788*X[s3]
+93.64506374942067*X[s4]
+505.8647575098491*X[s5]
+205.05717667186582*X[s6]
```

用一個含有bp的資料 vs 不含有bp的資料用 **Ridge regression + cross validation** 做訓練/預測，

比較兩者的 **MSE** 後會發現"有bp資料"所做的 **MSE** 會更小一些，

因次可得出 **“Average blood pressure” is an important factor for diabetes disease.**

""

除了 **Ridge** 以外其他迴歸方法如下 **(RidgeCV, GridSearchCV)**

"

In [21]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
ridgecv_reg = RidgeCV(alphas=[10**i for i in range(-10,1)]).fit(X_train, y_train)
y_pred = ridgecv_reg.predict(X_test)
MSE = mean_squared_error(y_test, y_pred)

print('MSE={MSE}\n'.format(MSE=MSE))

print('alpha={alpha}\n'.format(alpha=ridgecv_reg.alpha_))
print('coefficent={coef}\n'.format(coef=ridgecv_reg.coef_))
print('intercept={inter}\n'.format(inter=ridgecv_reg.intercept_))
print('model:Y={inter}'.format(inter=ridgecv_reg.intercept_), end="")
for i, name in enumerate(X.columns):
    print('+{coef}*X[{name}]\n'.format(coef=ridgecv_reg.coef_[i], name=name), end="")
```

In [21]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
ridgecv_reg = RidgeCV(alphas=[10**i for i in range(-10,1)]).fit(X_train, y_train)
y_pred = ridgecv_reg.predict(X_test)
MSE = mean_squared_error(y_test, y_pred)

print('MSE={MSE}\n'.format(MSE=MSE))

print('alpha={alpha}\n'.format(alpha=ridgecv_reg.alpha_))
print('coefficent={coef}\n'.format(coef=ridgecv_reg.coef_))
print('intercept={inter}\n'.format(inter=ridgecv_reg.intercept_))
print('model:Y={inter}'.format(inter=ridgecv_reg.intercept_), end="")
for i, name in enumerate(X.columns):
    print('+{coef}*X[{name}]\n'.format(coef=ridgecv_reg.coef_[i], name=name), end="")
```

```
MSE=2831.232290853817

alpha=0.1

coefficent=[   3.08183301 -196.3068712   529.43698456  306.05633085  -53.73080856
  -88.67125767 -174.97880268  133.27318777  402.9734622    78.46269483]

intercept=151.45196351013487

model:Y=151.45196351013487+3.0818330116414763*X[age]
+-196.30687119958577*X[sex]
+529.4369845637453*X[bmi]
+306.05633084923454*X[bp]
+-53.730808559008416*X[s1]
+-88.67125767121702*X[s2]
+-174.97880268226453*X[s3]
+133.2731877682104*X[s4]
+402.9734621956147*X[s5]
+78.46269482961065*X[s6]
```

In [22]:
```
parameter = {'alpha':[10**i for i in range(-10,1)]}
GSCV_reg = GridSearchCV(estimator=Ridge(), param_grid=parameter)
GSCV_reg.fit(X_train, y_train)
y_pred = GSCV_reg.predict(X_test)
MSE = mean_squared_error(y_test, y_pred)

print('MSE={MSE}\n'.format(MSE=MSE))

print('alpha={alpha}\n'.format(alpha=GSCV_reg.best_params_['alpha']))
print('coefficent={coef}\n'.format(coef=GSCV_reg.best_estimator_.coef_))
print('intercept={inter}\n'.format(inter=GSCV_reg.best_estimator_.intercept_))
print('model:Y={inter}'.format(inter=GSCV_reg.best_estimator_.intercept_), end="")
for i, name in enumerate(X.columns):
    print('+{coef}*X[{name}]\n'.format(coef=GSCV_reg.best_estimator_.coef_[i], name=name), end="")
```

```
MSE=2831.2322908538144

alpha=0.1

coefficent=[   3.08183301 -196.3068712   529.43698456  306.05633085  -53.73080856
  -88.67125767 -174.97880268  133.27318777  402.9734622    78.46269483]

intercept=151.4519635101349

model:Y=151.4519635101349+3.0818330116417285*X[age]
+-196.30687119959097*X[sex]
+529.436984563743*X[bmi]
+306.05633084923033*X[bp]
+-53.73080855900817*X[s1]
+-88.6712576712179*X[s2]
+-174.97880268226143*X[s3]
+133.27318776820636*X[s4]
+402.97346219561194*X[s5]
+78.46269482960702*X[s6]
```