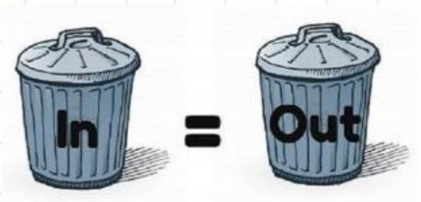
Python for Data Analysis
資料探勘

講者:楊翔斌
n07061033@mail.ncku.edu.tw

大綱

1. 一般方法簡述
2. 線性模型
3. 支援向量機
4. 決策樹

一般方法簡述



數據預整理

↓

數據分為訓練集、測試集及驗證集

↓

選擇模型及適當參數

↓

以訓練集建立模型

↓

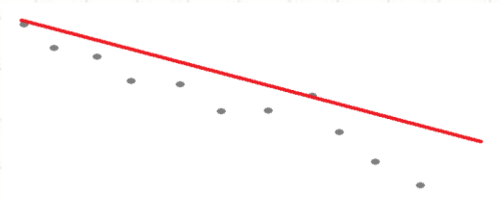
以測試集帶入模型測試準確性

將數據分成訓練及測試，會再分出驗證集驗證測試集的準確性

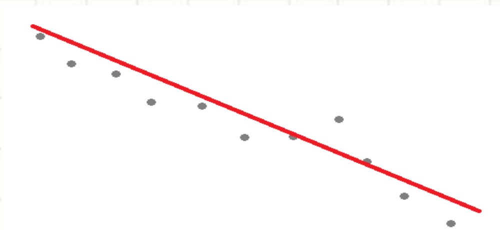
測試及驗證集驗證該模型，檢視fitting效果(overfitting或fitting不夠)

一般方法簡述

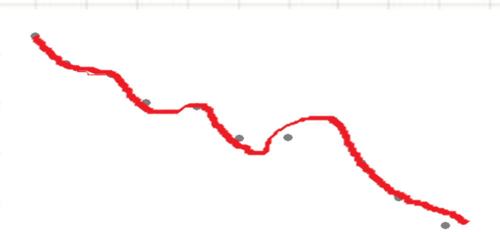
overfitting & under fitting



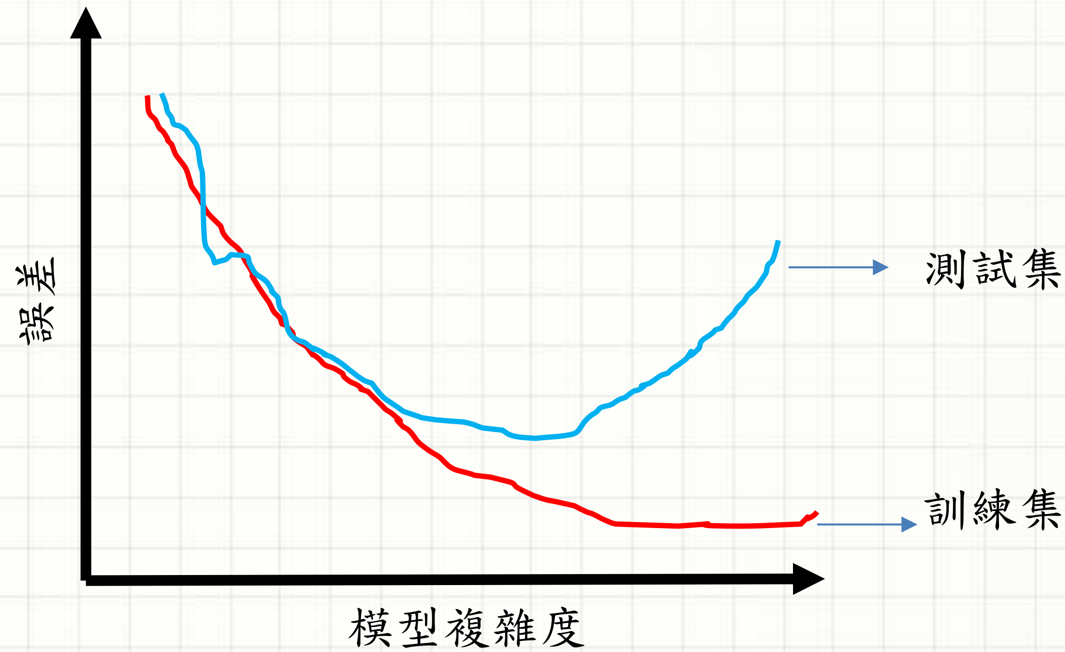
under fitting



fitting



over fitting



線性模型

模型形式

單變數回歸

$$y = \beta_0 + \beta_1 x_1 + \epsilon$$

依變數 y軸截距 直線斜率 獨立變數 誤差

多變數回歸

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \epsilon$$

依變數 y軸截距 直線斜率 獨立變數 直線斜率 獨立變數 誤差

線性模型

優點:操作容易、簡單易了解

缺點:模型過於簡單，預測結果不準

學習重點:評估模型準確率、
選擇適當變數
避免過度學習

線性模型

Code:

```
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import summary_table
from sklearn.cross_validation import #用於分割資料
feature_cols = ["blood pressure","specific gravity","blood glucose random","packed cell volume"]
x=sm.add_constant(data1_new[feature_cols])
y=data1_new['age']
X_train,X_test, y_train, y_test = train_test_split(x, y,
test_size = 0.25, random_state = 2018)#將數據分為訓練及待預測數集
regr=sm.OLS(y_train,X_train)#建立線性模型
res=regr.fit()
res.summary()
predata=res.predict(X_test)#以X_test數據帶入模型作預測
```

線性模型

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    age   R-squared:                       0.126
Model:                            OLS   Adj. R-squared:                  0.114
Method:                 Least Squares   F-statistic:                     10.28
Date:                Sun, 12 Aug 2018   Prob (F-statistic):           8.68e-08
Time:                        11:26:42   Log-Likelihood:                -1199.2
No. Observations:                 290   AIC:                             2408.
Df Residuals:                     285   BIC:                             2427.
Df Model:                           4
Covariance Type:            nonrobust
===========================================================================================
                              coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------------
const                    -101.9443    207.201     -0.492      0.623    -509.784     305.895
blood pressure              0.0869      0.073      1.192      0.234      -0.057       0.230
specific gravity          149.5347    205.495      0.728      0.467    -254.945     554.015
blood glucose random        0.0584      0.015      3.941      0.000       0.029       0.088
packed cell volume         -0.3441      0.129     -2.666      0.008      -0.598      -0.090
==============================================================================
Omnibus:                        9.024   Durbin-Watson:                   1.976
Prob(Omnibus):                  0.011   Jarque-Bera (JB):                8.953
Skew:                          -0.409   Prob(JB):                       0.0114
Kurtosis:                       3.267   Cond. No.                     5.86e+04
==============================================================================
```

線性模型

Code:

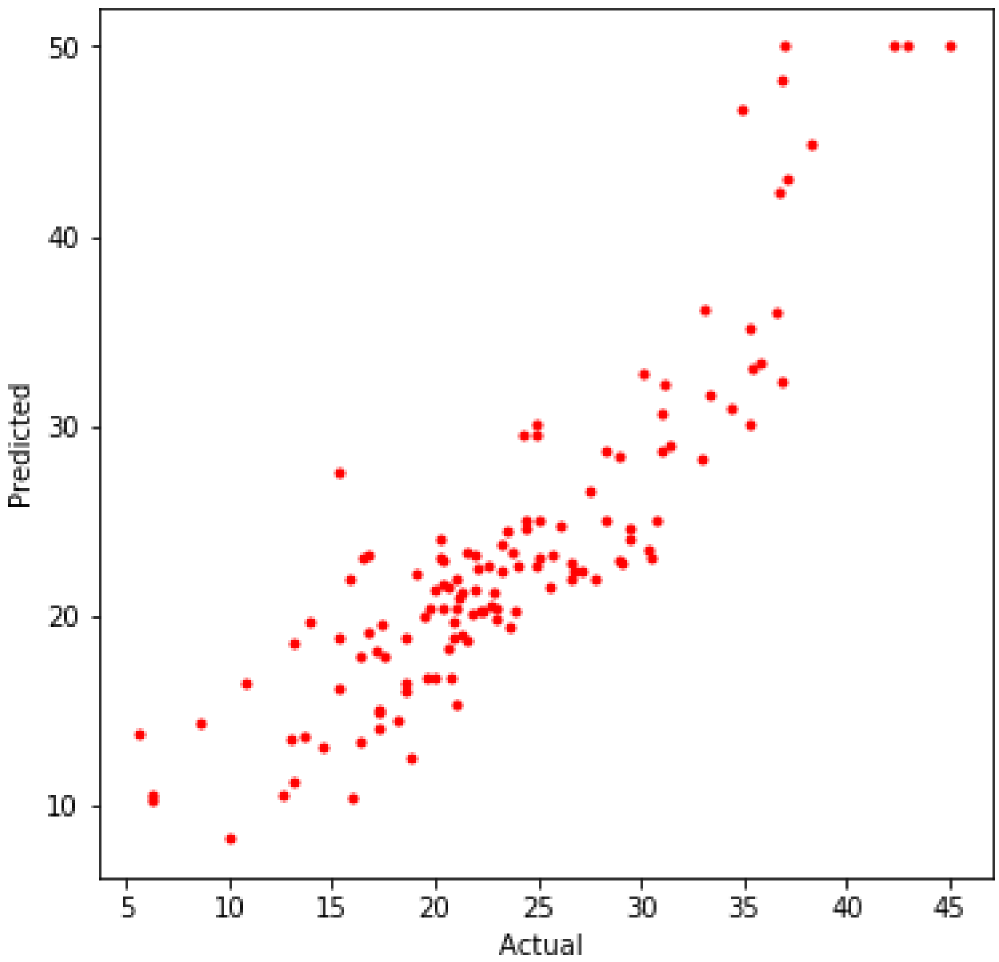
```
import statsmodels.api as sm
import pandas as pd
from statsmodels.stats.outliers_influence import summary_table
from sklearn.cross_validation import train_test_split#用於分割資料
from sklearn import datasets #呼叫內建資料集
boston = datasets.load_boston()
y = boston.target
newbo=boston.data
colu=boston.feature_names
newboston=pd.DataFrame(newbo,columns=colu)#對數據集欄命名
feature_cols = [colu]
x=sm.add_constant(newboston[feature_cols[0]])
X_train,X_test, y_train, y_test = train_test_split(x, y,
test_size = 0.25, random_state = 2018)
regr=sm.OLS(y_train,X_train)
res=regr.fit()
res.summary()
pre=res.predict(X_test)
```

線性模型

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.728
Model:                            OLS   Adj. R-squared:                  0.719
Method:                 Least Squares   F-statistic:                     75.27
Date:                Sun, 12 Aug 2018   Prob (F-statistic):           9.67e-95
Time:                        17:51:38   Log-Likelihood:                -1137.3
No. Observations:                 379   AIC:                             2303.
Df Residuals:                     365   BIC:                             2358.
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         39.6419      6.068      6.533      0.000      27.709      51.574
CRIM          -0.0827      0.044     -1.864      0.063      -0.170       0.005
ZN             0.0489      0.017      2.935      0.004       0.016       0.082
INDUS          0.0060      0.073      0.082      0.935      -0.137       0.149
CHAS           3.3764      1.081      3.123      0.002       1.250       5.503
NOX          -18.8355      4.469     -4.215      0.000     -27.624     -10.047
RM             3.4751      0.508      6.846      0.000       2.477       4.473
AGE            0.0093      0.016      0.579      0.563      -0.022       0.041
DIS           -1.5675      0.245     -6.387      0.000      -2.050      -1.085
RAD            0.2883      0.081      3.559      0.000       0.129       0.448
TAX           -0.0123      0.004     -2.750      0.006      -0.021      -0.004
PTRATIO       -0.9262      0.159     -5.812      0.000      -1.240      -0.613
B              0.0081      0.003      2.584      0.010       0.002       0.014
LSTAT         -0.5750      0.062     -9.280      0.000      -0.697      -0.453
==============================================================================
Omnibus:                      133.325   Durbin-Watson:                   2.104
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              548.979
Skew:                           1.496   Prob(JB):                    6.18e-120
Kurtosis:                       8.081   Cond. No.                     1.48e+04
```

P-value:越小表示此參數越重要

線性模型



MSE=17

線性模型

廣義線性模型:

- 使用時機:獨立變數(x)可以為連續變數及類別變數。
  依變數(y)亦可以為連續變數及類別變數。
  所建立模型可用於作類別判斷的預測。

- 類別變數需要轉成啞變數才能做fitting。

- 類別變數如有n個，轉成啞變數後只需用n-1個變量作fitting即可。

線性模型

Code:利用腎臟病數據

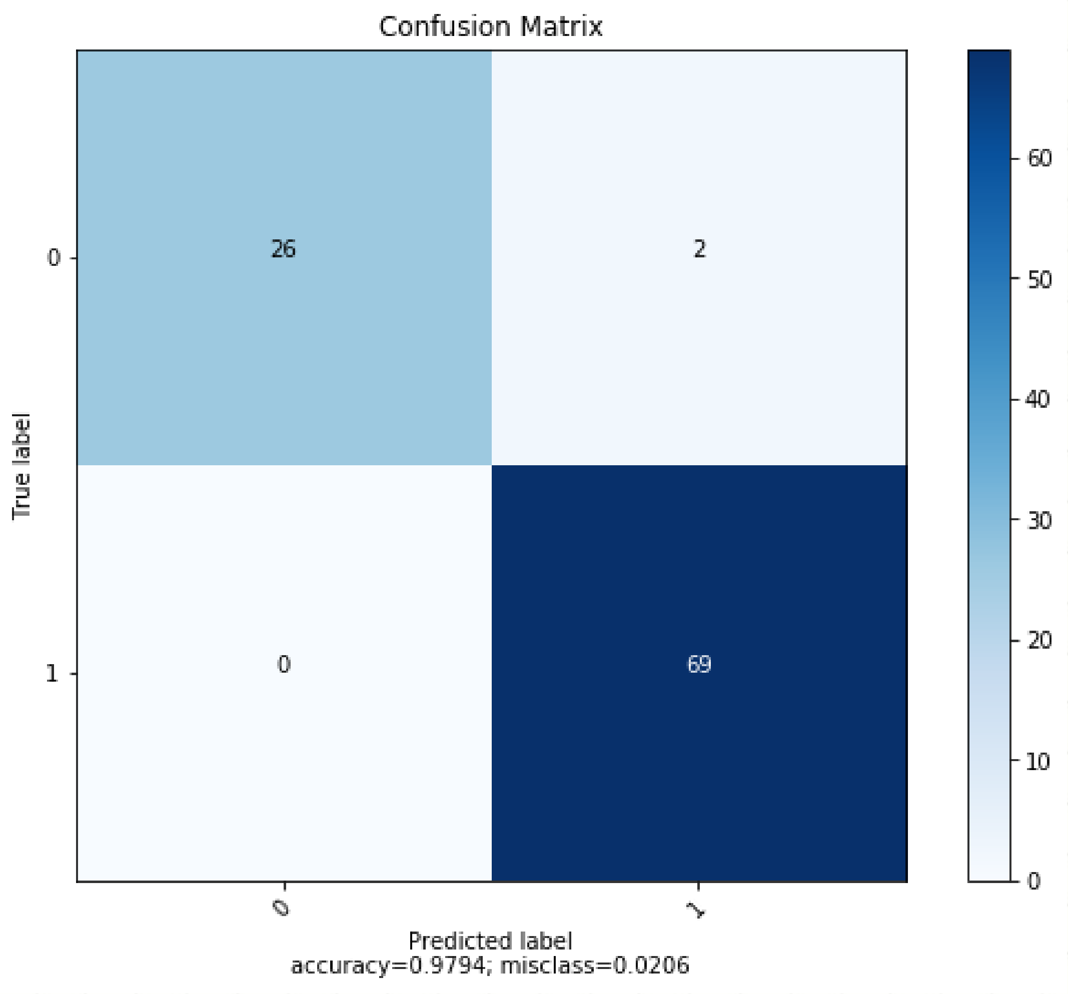
請參閱
https://github.com/jasonfghx/py/tree/master/python_for_class/GLM_example
中的code

線性模型

	coef	std err	z	P>|z|	[0.025	0.975]
const	31.0078	3.567	8.692	0.000	24.016	38.000
age	-0.0010	0.001	-0.934	0.350	-0.003	0.001
blood pressure	0.0018	0.001	1.510	0.131	-0.001	0.004
specific gravity	-28.9083	3.543	-8.160	0.000	-35.852	-21.965
blood glucose random	1.881e-05	0.000	0.063	0.949	-0.001	0.001
blood urea	-0.0013	0.001	-2.187	0.029	-0.002	-0.000
serum creatinine	-0.0013	0.006	-0.237	0.813	-0.012	0.010
sodium	-0.0035	0.003	-1.374	0.169	-0.009	0.002
potassium	-0.0037	0.006	-0.589	0.556	-0.016	0.009
hemoglobin	-0.0405	0.012	-3.264	0.001	-0.065	-0.016
packed cell volume	-0.0014	0.004	-0.337	0.736	-0.009	0.007
white blood cell count	8.885e-07	6.41e-06	0.139	0.890	-1.17e-05	1.35e-05
red blood cell count	-0.0161	0.032	-0.503	0.615	-0.079	0.047
albumin_1	0.3788	0.058	6.513	0.000	0.265	0.493
albumin_2	0.3992	0.066	6.020	0.000	0.269	0.529
albumin_3	0.3075	0.069	4.484	0.000	0.173	0.442
albumin_4	0.2927	0.092	3.182	0.001	0.112	0.473
albumin_5	0.4552	0.303	1.501	0.133	-0.139	1.050
sugar_1	-0.0702	0.069	-1.020	0.308	-0.205	0.065
sugar_2	0.1350	0.098	1.375	0.169	-0.057	0.328
sugar_3	0.0651	0.093	0.697	0.486	-0.118	0.248
sugar_4	0.0848	0.106	0.796	0.426	-0.124	0.293
sugar_5	0.1300	0.188	0.692	0.489	-0.238	0.498
red_blood_cells_2	-0.0091	0.048	-0.189	0.850	-0.103	0.085
pus_cell_2	-0.0591	0.054	-1.088	0.277	-0.166	0.047
pus_cell_clumps_2	-0.0791	0.063	-1.260	0.208	-0.202	0.044
bacteria__2	0.0164	0.086	0.191	0.848	-0.152	0.185
hypertension__2	-0.0847	0.052	-1.621	0.105	-0.187	0.018
diabetes_mellitus_1	0.0195	0.257	0.076	0.939	-0.485	0.524
diabetes_mellitus_2	-0.0896	0.260	-0.345	0.730	-0.599	0.420
coronary_artery_disease_2	0.0818	0.069	1.187	0.235	-0.053	0.217
appetite__2	0.0858	0.046	1.845	0.065	-0.005	0.177
pedal_edema_2	-0.0016	0.051	-0.031	0.975	-0.101	0.098
anemia__2	0.0078	0.057	0.137	0.891	-0.104	0.120

線性模型

評估類別變數預測準確性會用到混淆矩陣:



線性模型

Lasso回歸:

- 建置模型時常會有overfitting的情況發生，lasso為一種減少overfitting的方法。
- lasso會給予變數權重(係數)及限制模型複雜度。
- lasso常用於變數之挑選。

線性模型

Code:

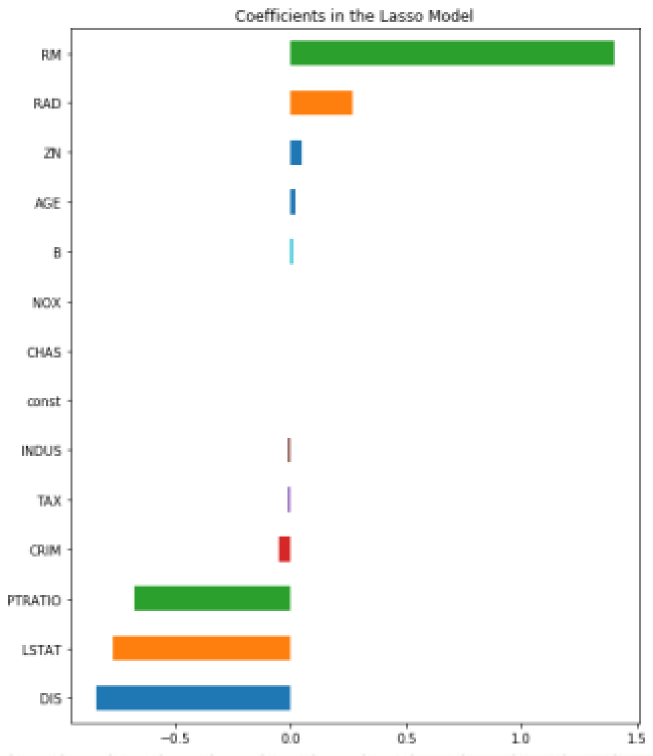
```
from sklearn.cross_validation import train_test_split
from sklearn import datasets
import matplotlib as plt
from sklearn.linear_model import Lasso
boston = datasets.load_boston()
y = boston.target
newbo=boston.data
colu=boston.feature_names
newboston=pd.DataFrame(newbo,columns=colu)
X_train,X_test, y_train, y_test = train_test_split(newboston, y, test_size =
0.25, random_state = 2018)
lasso = Lasso()
model=lasso.fit(X_train, y_train)
a=model.predict(X_test)
import matplotlib.pyplot as plt
plt.scatter(a,y_test,marker="^",c="g")
mse = np.mean((a- y_test) ** 2)
```

線性模型

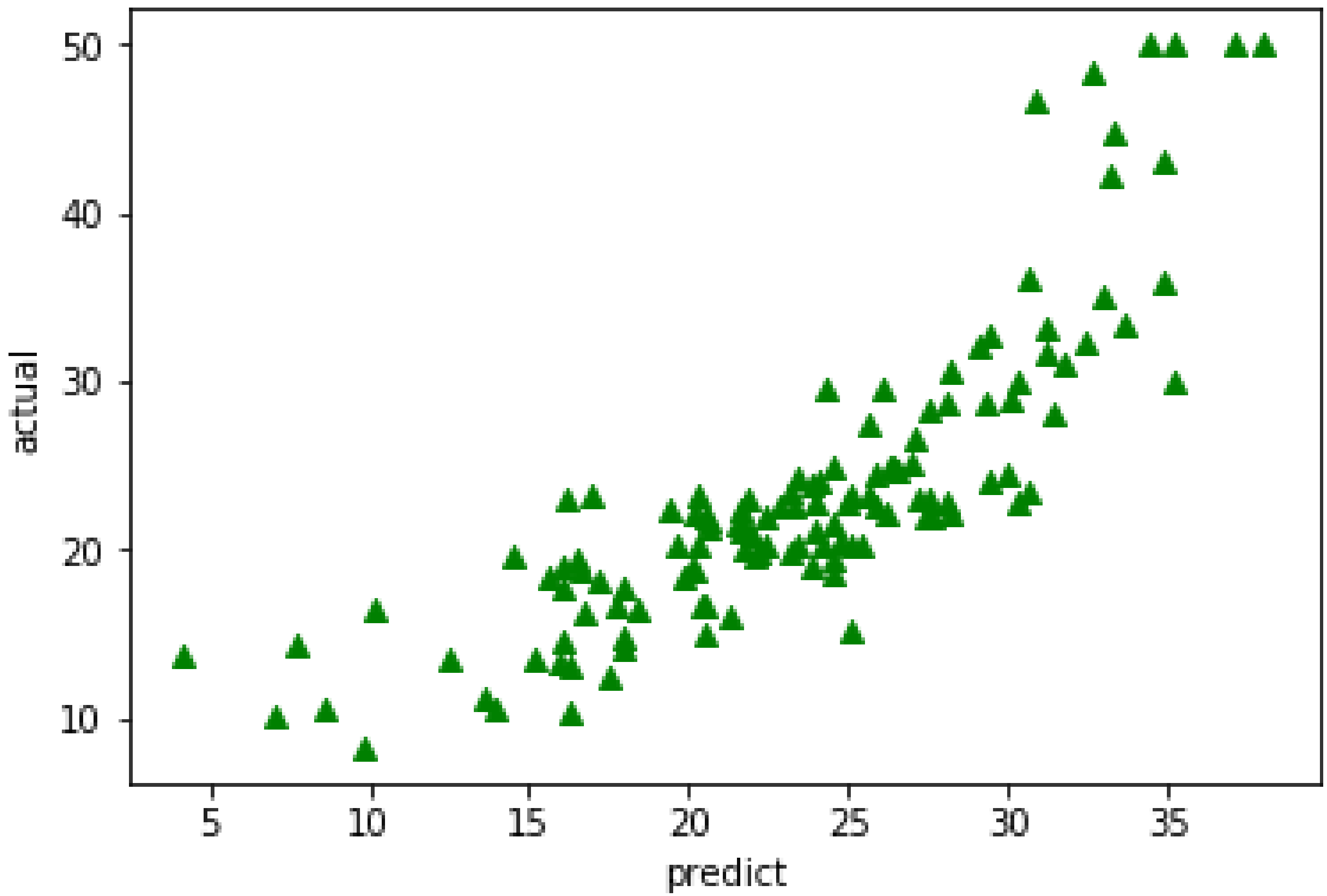
lasso回歸取得之係數，越接近0表示越不重要

```
In [19]: lasso.coef_
Out[19]:
array([-0.040704  ,  0.04971457, -0.        ,  0.        , -0.        ,
        0.57714858,  0.03148448, -0.68241987,  0.26664202, -0.01602919,
       -0.65096563,  0.00696196, -0.828762  ])
```

```
coef = pd.Series(lassocv.coef_, index = X_train.columns)
imp_coef = coef.sort_values()
plt.rcParams['figure.figsize'] = (8.0, 10.0)
imp_coef.plot(kind = "barh")
plt.title("Coefficients in the Lasso Model")
```



線性模型



MSE=23.5

線性模型

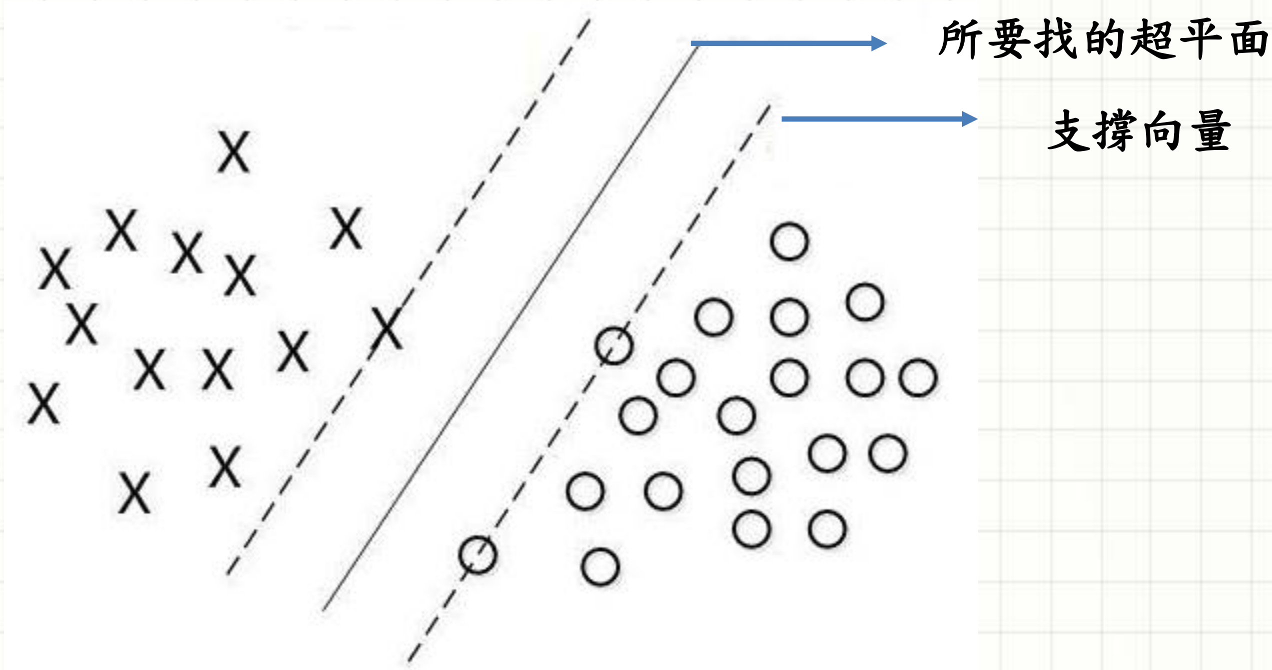
小結:

- lasso可以協助萃取重要參數。
- 精準度與運算時間的權衡

支援向量機

支援向量機(SVM)

- 原理:找到一超平面(向量)能夠將數據分得越開越好。



支援向量機

支援向量機(SVM)重要參數:

- C值:C值越大所允許誤差越小，模型所學的數據越準，但對於新樣本得預測不一定很準，也就是出現overfitting。
  C值越小所允許誤差越大，模型在分類上容易錯，新樣本得預測也可能出錯。

- gamma值:主要用於kernel function中poly、rbf、sigmoid，簡而言之是決定資料在特徵空間中分布狀況的參數。

支援向量機

Code:

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn import metrics
from sklearn.metrics import confusion_matrix
df = pd.read_csv('voice.csv')
X=df.iloc[:, :-1]#擷取變數項
Y=df.iloc[:, -1:] #擷取待預測項
gender_encoder = LabelEncoder()
y = gender_encoder.fit_transform(Y)#將文字label轉為數字
scaler = StandardScaler()#標準化
scaler.fit(X)
X = scaler.transform(X)
#接續
```

支援向量機

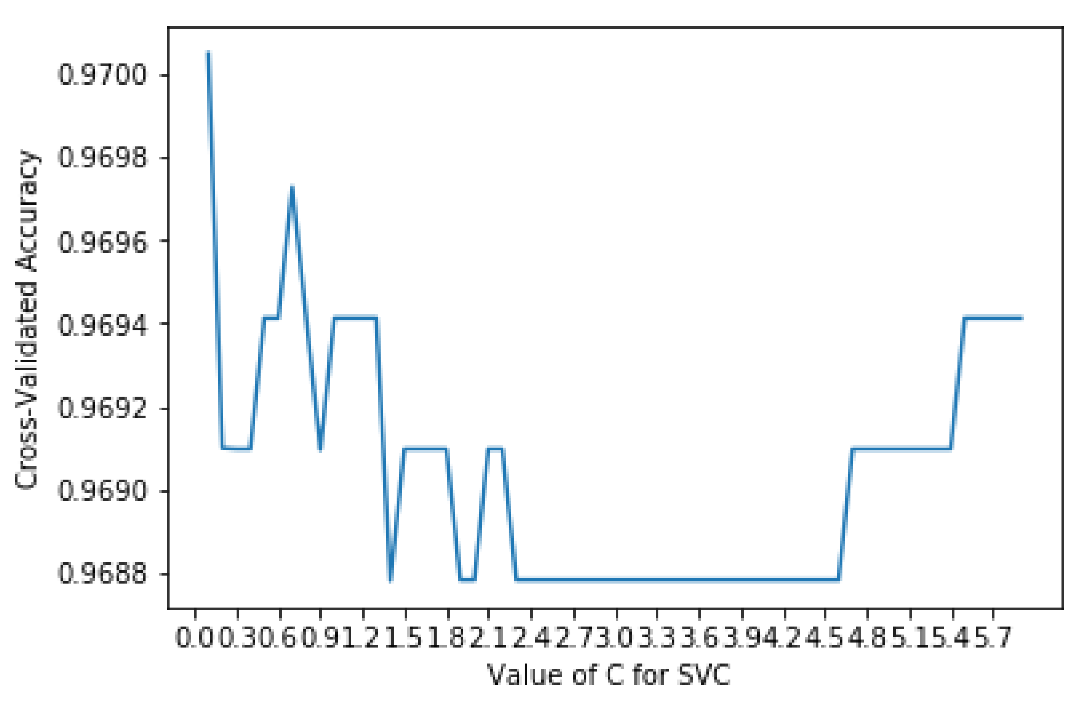
Code:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=1)#分為train test項
svc=SVC(kernel='linear')
#SVC內的kernel可以為linear、poly、rbf、sigmoid
svc.fit(X_train,y_train)
y_pred=svc.predict(X_test)
print('Accuracy Score:')
print(metrics.accuracy_score(y_test,y_pred))
confusion_matrix(y_pred, y_test)
############tune 參數 #################
#針對C值跑loop求最佳值
C_range=list(np.arange(0.1,6,0.1))
acc_score=[]
for c in C_range:
    svc = SVC(kernel='linear', C=c)
    scores = cross_val_score(svc, X, y, cv=10, scoring='accuracy')
    acc_score.append(scores.mean( ))
#接續
```

支援向量機

Code:

```
plt.plot(C_range,acc_score)
plt.xticks(np.arange(0.0,6,0.3))
plt.xlabel('Value of C for SVC ')
plt.ylabel('Cross-Validated Accuracy')
```



由此可以決定C=0.1能夠使模型有較佳準確率

此法非常耗時

支援向量機

運用grid search找尋最佳參數(*大數據競賽常用*)

Code:

```
from sklearn.grid_search import GridSearchCV
tuned_parameters =
{ 'C': (np.arange(0.1,1,0.1)) , 'kernel': ['linear'],
  'C': (np.arange(0.1,1,0.1)) , 'gamma': [0.01,0.02,0.03,0.04,0.05], 'kernel': ['rbf'],
'degree': [2,3,4] ,'gamma':[0.01,0.02,0.03,0.04,0.05], 'C':(np.arange(0.1,1,0.1)) ,
'kernel':['poly'] }
model_svm = GridSearchCV(svm_model,
tuned_parameters,cv=10,scoring='accuracy')

model_svm.fit(X_train, y_train)
print(model_svm.best_score_)
print(model_svm.best_params_)
```

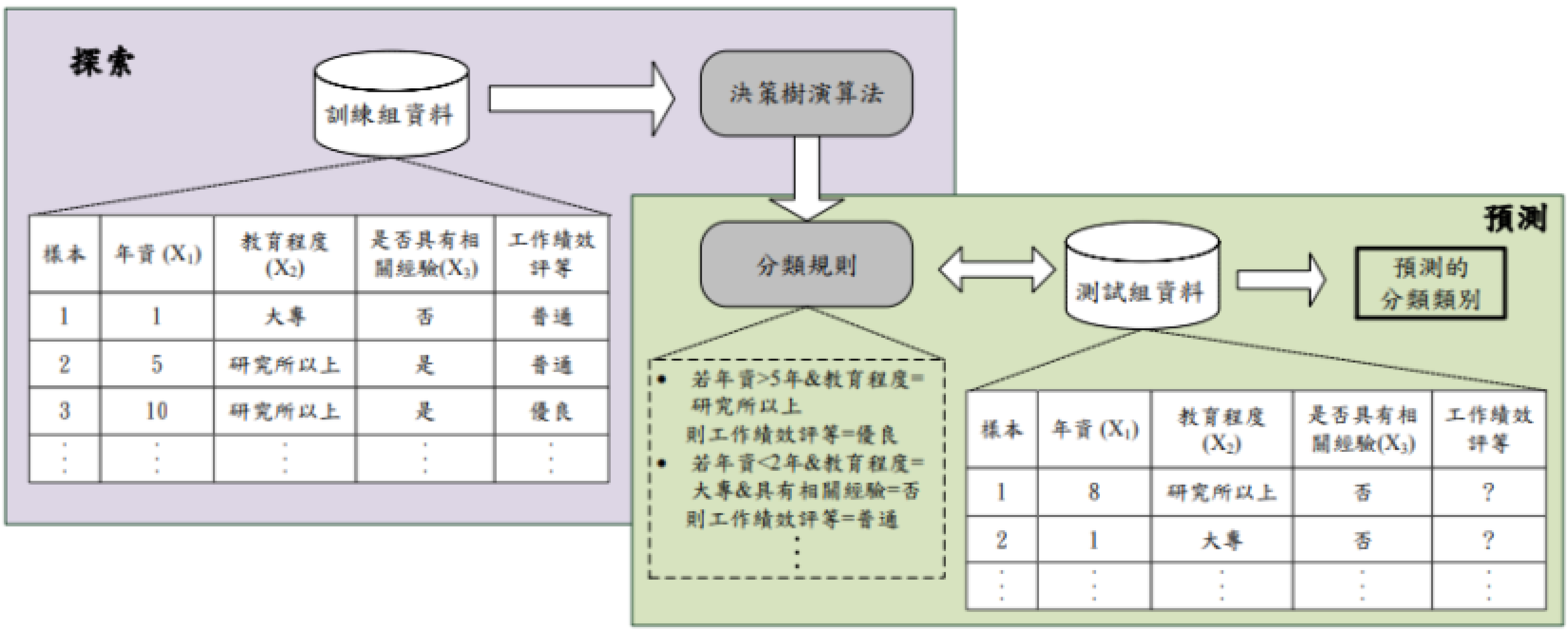
```
In [197]: print(model_svm.best_score_)
0.9569850039463299

In [198]: print(model_svm.best_params_)
{'C': 0.9, 'degree': 3, 'gamma': 0.05, 'kernel': 'poly'}
```

決策樹

具備特徵擷取和描述之功能，將變數透過模型計算選擇分支的個數及方式，以樹枝狀形式呈現出分類的規則。

目的:找規則及預測



決策樹

範例:

height	weight	hair length	voice	gender
180	85	15	0	man
177	59	42	0	woman
136	55	35	1	woman
174	69	65	0	man
141	60	28	1	woman
170	66	60	1	woman

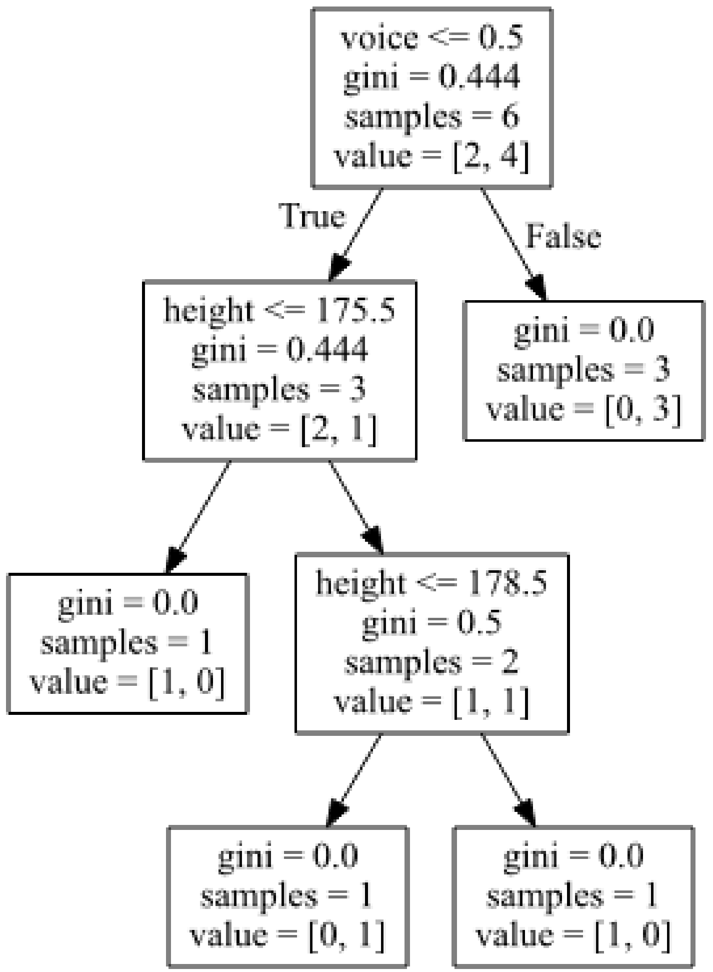
用於建模的參數　　要預測的類別

決策樹

Code:

```
from sklearn.tree import export_graphviz
import graphviz
from sklearn import tree
X2 = [ [180,85, 15,0],
       [177,60, 42,0],
       [136,55, 35,1],
       [174,59, 65,0],
       [141,60, 28,1],
       [170,66,60,1]]
Y2 = ['man', 'woman', 'woman', 'man', 'woman', 'woman']
data_feature_names = [ 'height',"weight", 'hair length', 'voice' ]
clf = tree.DecisionTreeClassifier()#建立決策樹模型
clf = clf.fit(X2,Y2)
export_graphviz(clf,feature_names=data_feature_names, out_file="mytree.dot")
#輸出決策樹模型文字敘述檔
with open("mytree.dot") as f:
   dot_graph = f.read()
graphviz.Source(dot_graph) #決策樹文字視覺化
```

決策樹



模型屬性參數:

- **clf.classes_：顯示類別實際名稱**

  Out[41]: array(['man', 'woman'], dtype='<U5')

- **clf.feature_importances_：表示變數重要性**

  Out[45]: array([0.5, 0. , 0. , 0.5])

- **clf.max_features_：表示用到多少變數**

  Out[81]: 4

- **clf. max_depth：可以指定最大深度**
- **clf.min_samples_split：可指定分解內部結點時最少樣本**

決策樹所能給的資訊:

- **分類的邏輯**
- **If-then的資訊:if voice<=0.5 AND h<=175.5 then man**

決策樹

建置步驟:

資料準備及處理

↓

決策樹建模

↓

決策樹修剪

↓

用數據驗證效果

↓

獲取規則

決策樹

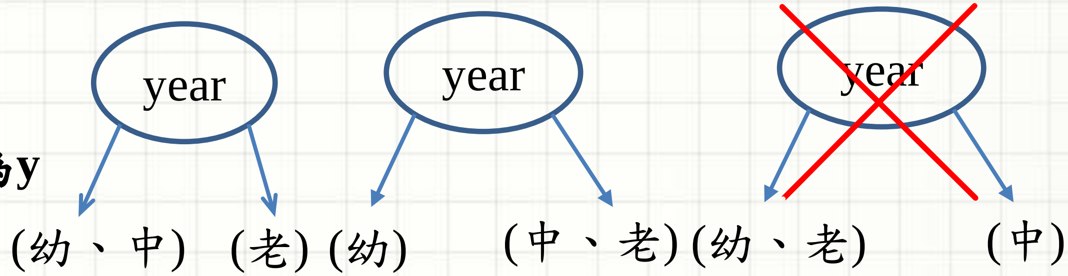
資料準備及處理:

一、預備分析的資料分為兩種參數:

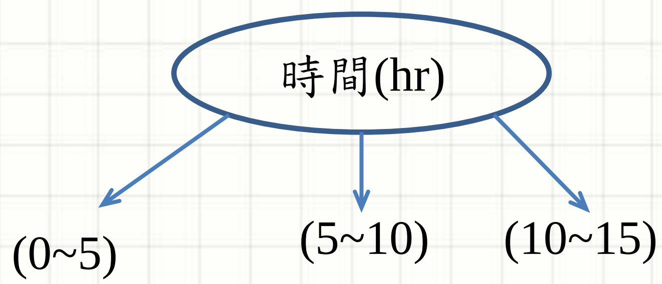
1. 目標變數(y)，可以為二元或多元。
2. 建置模型用到的參數(x)，也稱分支變數。

二、屬性:

順序屬性:將順序變成群組做為y



連續屬性:若是連續變數可以用區間做分割



決策樹

決策樹建模:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import metrics
df = pd.read_csv('voice.csv')
X=df.iloc[:, :-1]
Y=df.iloc[:, -1:]
gender_encoder = LabelEncoder()
y = gender_encoder.fit_transform(Y)
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=1)
```

續

決策樹

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(criterion = 'entropy', random_state=0)
tree.fit(X_train,y_train)
pretree=tree.predict(X_test)
print(metrics.accuracy_score(y_test,pretree))
#準確率 = 0.9652

tree = DecisionTreeClassifier(criterion = 'gini', random_state=0)
tree.fit(X_train,y_train)
pretree=tree.predict(X_test)
print(metrics.accuracy_score(y_test,pretree))
#準確率 = 0.979
```

更強大的模型是由很多決策樹組成的***隨機森林***

決策樹

隨機森林:

- **為集成學習中的一種方法，有三個臭皮匠勝過一個諸葛亮的概念。主要是集合數個模型，截長補短結合成一個模型。**
- **隨機森林顧名思義，做出數個決策樹，將其結果做投票，得出最高票的類別。**

Code:

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor( n_estimators = 1000, random_state = 42,
max_depth=100)
# max_depth 限制數的深度，若不限制則會一直分到一類只有一個，overfitting
#n_estimators表示森林中要建置樹的數量

rf.fit(X_train,y_train)
y_pred = rf.predict_(X_test)
a=pd.DataFrame(y_pred)
a=round(a)
print(metrics.accuracy_score(y_test,a))
#準確率 = 0.97，因為數據集本身太好，所以差異不大
```

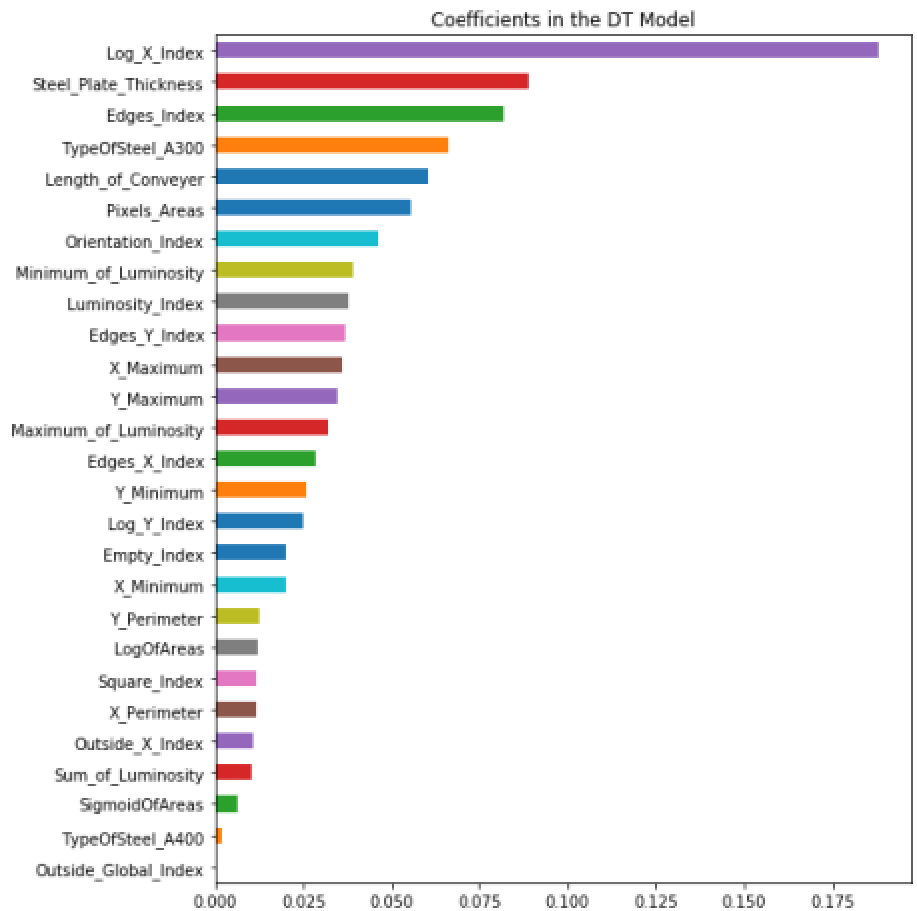
決策樹

演練(以鋼缺陷數據為例):

https://github.com/jasonfghx/py/edit/master/python_for_class/tree/fault

1.以決策樹模型算出的預測值準確率為**0.69**

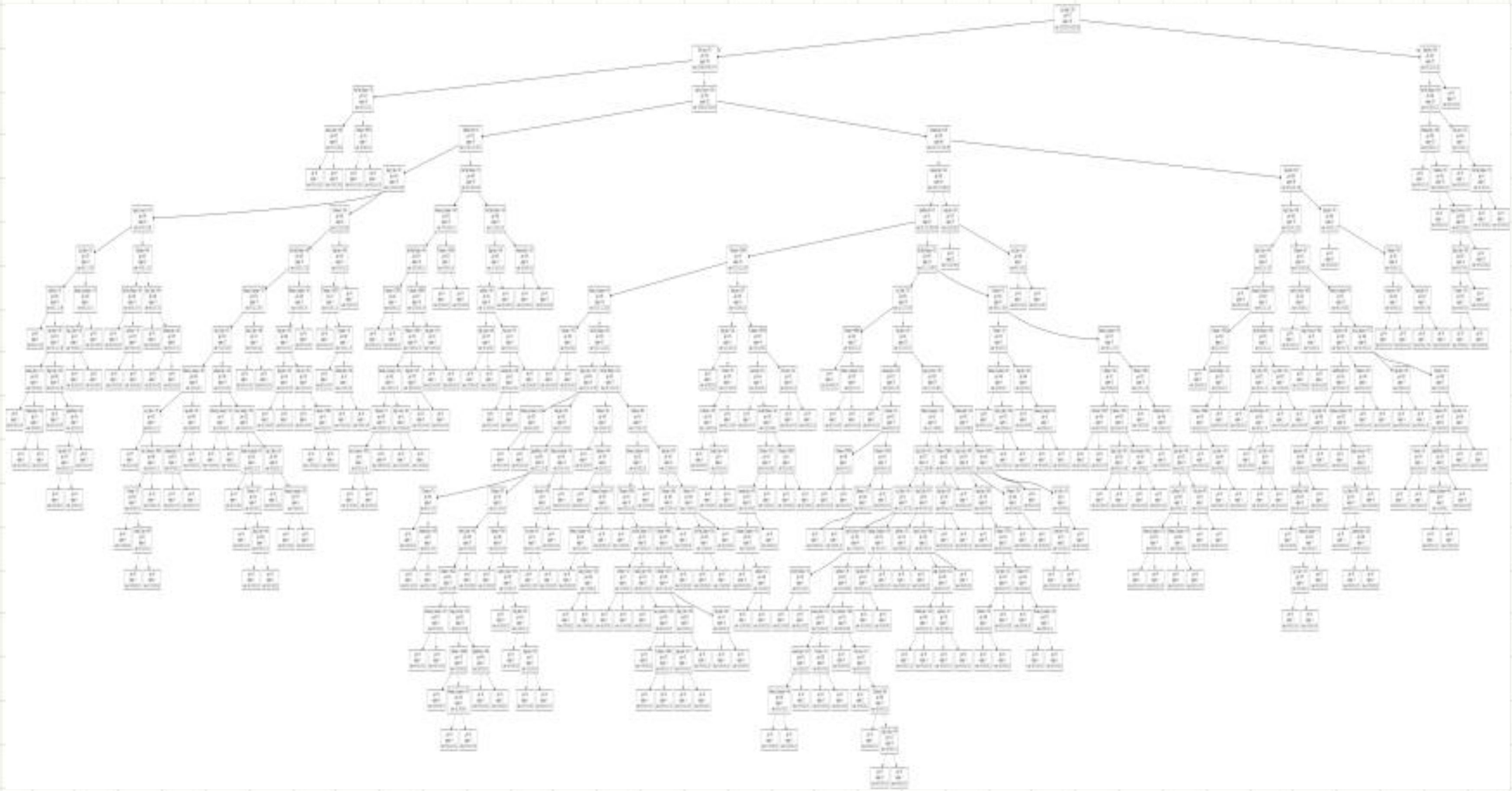
2.表示重要參數



在單棵數情況
此參數最重要

決策樹

所畫出的大型決策樹



決策樹

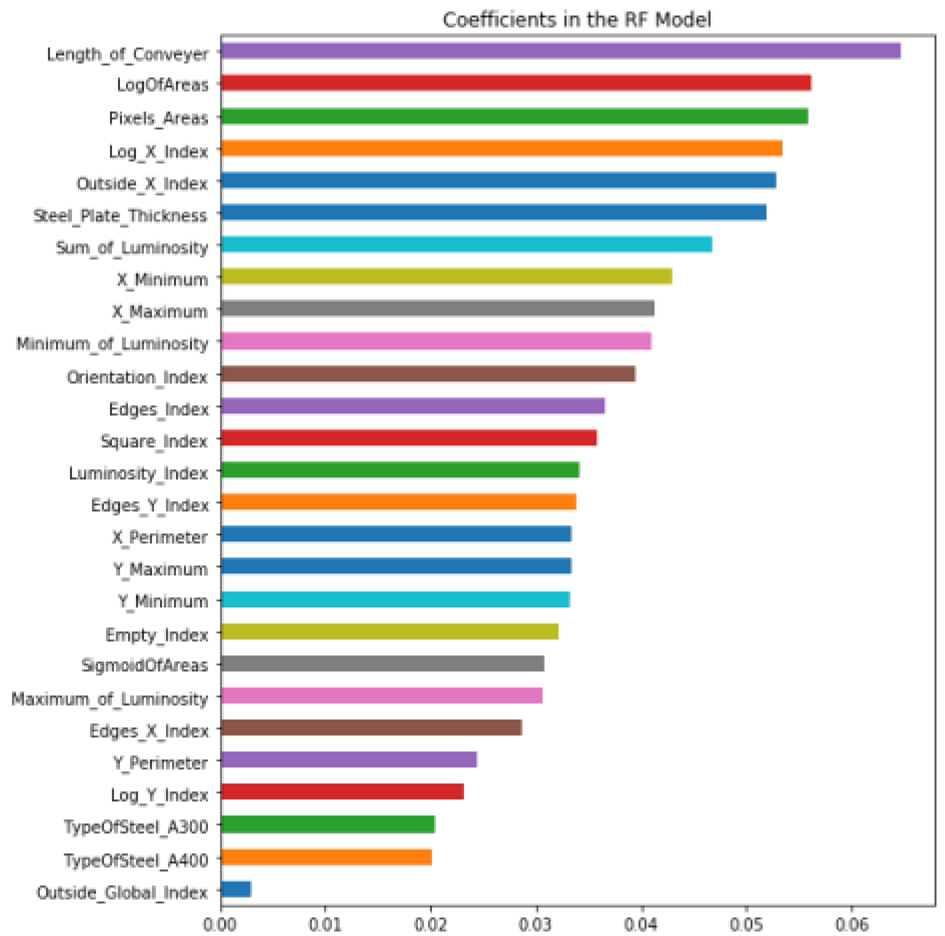
演練(以鋼缺陷數據為例):

1.以隨機森林模型算出的預測值準確率為0.79

```
In [377]: print(metrics.accuracy_score(y_test,y_pred1))
0.7917737789203085
```

2.表示重要參數

重要性參數是由多棵樹平均之結果



決策樹

運用grid search找尋最佳參數(*大數據競賽常用*)

Code:

from sklearn.grid_search import GridSearchCV
param_test1 = {'max_depth':list(range(3,5,2)), 'min_samples_split':list(range(50,60,20))}
#max_depth會測試3跟5，min_samples_split會測試50及60
param_test1 = {‘min_samples_split’:list(range(50,60,5))}#用此做測試
 # max_depth通常不會測試超過100
clf = GridSearchCV(RandomForestClassifier(n_estimators = 10000, random_state = 100), param_test1)
pre=y_train[“type”]#將y_train這個dataframe轉成series
clf.fit(X_train,pre)
clf.grid_scores_, clf.best_params_, clf.best_score_
#顯示出最佳參數(運行約2小時!!)

```
In [400]: clf.grid_scores_, clf.best_params_, clf.best_score_
Out[400]:
([mean: 0.72358, std: 0.02391, params: {'min_samples_split': 50},
  mean: 0.72036, std: 0.02467, params: {'min_samples_split': 55}],
 {'min_samples_split': 50},
 0.7235824742268041)
```