建立模型 - Python (2)

David Chiu

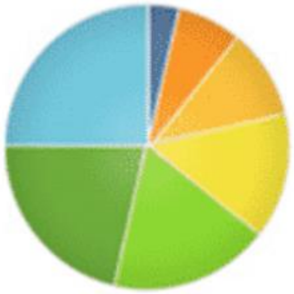
課程資料

- 所有課程補充資料、投影片皆位於
  - https://github.com/ywchiu/ctbcpy

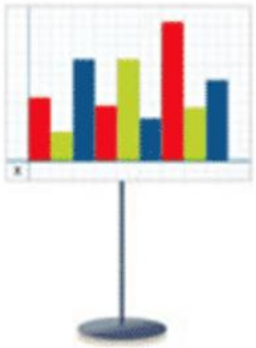
借貸俱樂部資料分析

Lending Club

Borrowers apply for loans.
Investors open an account.



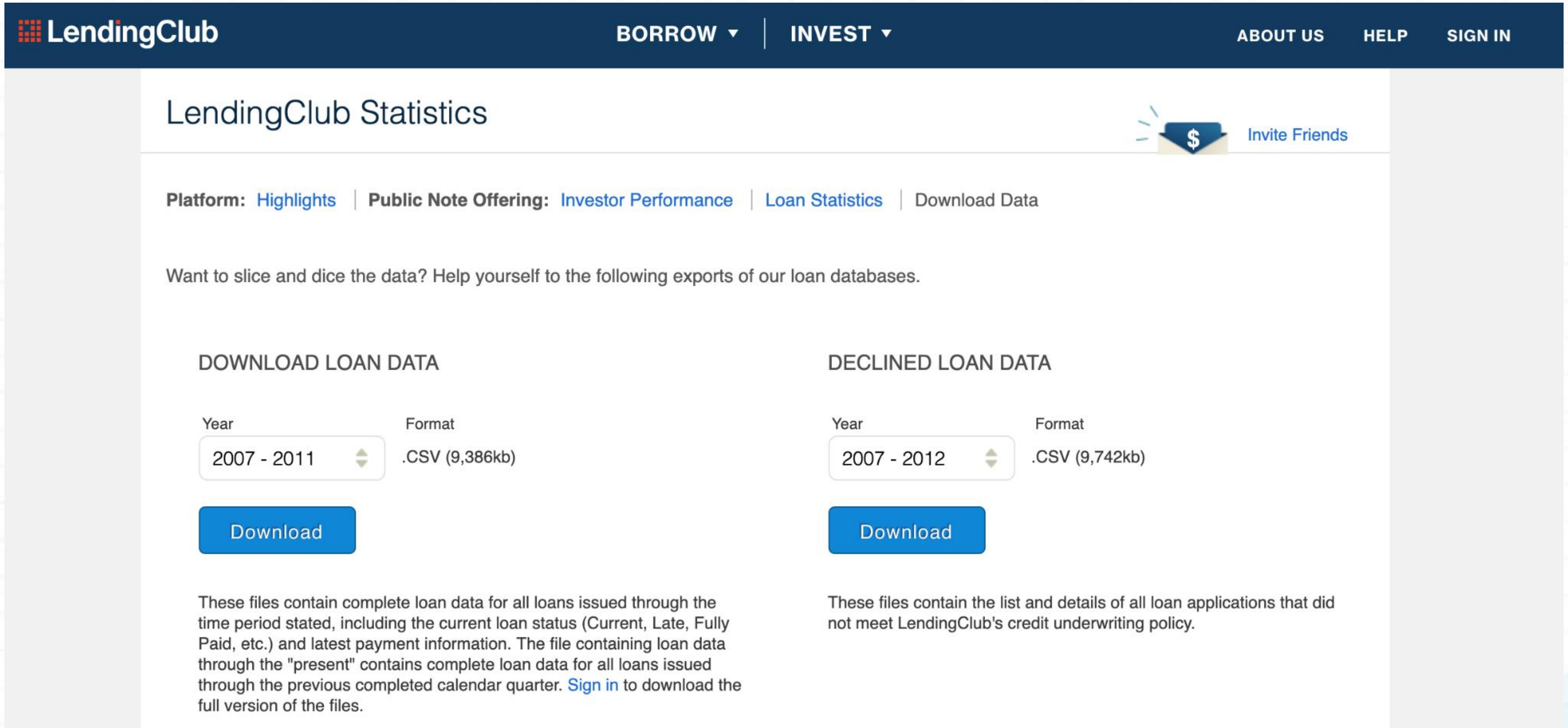
Borrowers get funded.
Investors build a portfolio.



Borrowers repay automatically.
Investors earn & reinvest.

借貸俱樂部資料

- https://www.lendingclub.com/info/download-data.action



資料讀取與清理

```
import pandas as pd

#讀取資料
dataset = pd.read_csv('LoanStats.csv')

#移除空白欄位
dataset = dataset.iloc[:,2:111]
empty_cols = [i for i in range(45,72)]
dataset = dataset.drop(dataset.columns[empty_cols],axis=1)
data_with_loanstatus_sliced = dataset[(dataset['loan_status']=="Fully Paid") | (dataset['loan_status']=="Charged Off")]

#轉換目標編碼
di = {"Fully Paid":0, "Charged Off":1}
Dataset_withBoolTarget= data_with_loanstatus_sliced.replace({"loan_status": di})
```

移除空白列與欄位

```
#移除空白列
dataset=Dataset_withBoolTarget.dropna(thresh = 340000,axis=1)
print("Current shape of dataset :",dataset.shape)

#移除欄位
del_col_names = ["delinq_2yrs",  "last_pymnt_d", "chargeoff_within_12_mths","delinq_amnt","emp_title", "term", "emp_title", "pymnt_plan","purpose","title", "zip_code", "verification_status", "dti","earliest_cr_line", "initial_list_status", "out_prncp",
"pymnt_plan", "num_tl_90g_dpd_24m", "num_tl_30dpd", "num_tl_120dpd_2m", "num_accts_ever_120_pd", "delinq_amnt",
"chargeoff_within_12_mths", "total_rec_late_fee", "out_prncp_inv", "issue_d"]
dataset = dataset.drop(labels = del_col_names, axis = 1)
print("Current shape of dataset :",dataset.shape)
```

篩選欄位

```
#篩選欄位
features = ['funded_amnt','emp_length','annual_inc','home_ownership','grade',
            "last_pymnt_amnt", "mort_acc", "pub_rec", "int_rate", "open_acc","num_actv_rev_tl",
            "mo_sin_rcnt_rev_tl_op","mo_sin_old_rev_tl_op","bc_util","bc_open_to_buy",
            "avg_cur_bal","acc_open_past_24mths",'loan_status'] #'sub_grade' #selecting final features #'addr_state''tax_liens',
Final_data = dataset[features] #19 features with target var
Final_data["int_rate"] = Final_data["int_rate"].apply(lambda x:float(x[:-1]) ) #reomving % sign, conv to float  - int_rate column
Final_data= Final_data.reset_index(drop=True)
print("Current shape of dataset :",Final_data.shape)
```

資料轉換

```
#Data encoding
Final_data['grade'] = Final_data['grade'].map({'A':7,'B':6,'C':5,'D':4,'E':3,'F':2,'G':1})
Final_data["home_ownership"] =
Final_data["home_ownership"].map({"MORTGAGE":6,"RENT":5,"OWN":4,"OTHER":3,"NONE":
2,"ANY":1})
Final_data["emp_length"] = Final_data["emp_length"].fillna('0')
Final_data["emp_length"] = Final_data["emp_length"].replace({'years':'','year':'','
':'','<':'','\+':'','n/a':'0'}, regex = True)
Final_data["emp_length"] = Final_data["emp_length"].apply(lambda x:int(x))
print("Current shape of dataset :",Final_data.shape)
Final_data.head()
```

使用平均值填補遺失值

```
Final_data.fillna(Final_data.mean(),inplace = True)
print("Current shape of dataset :",Final_data.shape)
```

將特徵值標準化

```
from sklearn import preprocessing,metrics
scl = preprocessing.StandardScaler()
fields = Final_data.columns.values[:-1]
data_clean = pd.DataFrame(scl.fit_transform(Final_data[fields]),
columns = fields)
data_clean['loan_status'] = Final_data['loan_status']
data_clean['loan_status'].value_counts()
```

合併清理過後的資料

```
loanstatus_0 = data_clean[data_clean["loan_status"]==0]
loanstatus_1 = data_clean[data_clean["loan_status"]==1]
subset_of_loanstatus_0 = loanstatus_0.sample(n=5500)
subset_of_loanstatus_1 = loanstatus_1.sample(n=5500)
data_clean = pd.concat([subset_of_loanstatus_1,
subset_of_loanstatus_0])
data_clean = data_clean.sample(frac=1).reset_index(drop=True)
print("Current shape of dataset :",data_clean.shape)
data_clean.head()
```

將資料區分為訓練與測試資料集

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data_clean.iloc[:,:-1], data_clean.iloc[:,-1], test_size=0.2, random_state=42)
```

Recursive Feature Elimination

```
from sklearn import linear_model,svm
from sklearn.feature_selection import RFE
# create the RFE model and select 3 attributes
clf_LR = linear_model.LogisticRegression(C=1e30)
clf_LR.fit(X_train,y_train)
rfe = RFE(clf_LR, 10)
rfe = rfe.fit(data_clean.iloc[:,:-1].values, data_clean.iloc[:,-1].values)
# summarize the selection of the attributes
print(rfe.support_)
print(rfe.ranking_)
```

PCA

```
from sklearn.decomposition import PCA
pca = PCA(n_components=10, whiten=True)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_
print('Expected Variance is '+ str(explained_variance))
```

資料篩選

```
features = ['funded_amnt','annual_inc','grade',"last_pymnt_amnt", "int_rate", "mo_sin_rcnt_rev_tl_op","mo_sin_old_rev_tl_op","bc_util","bc_open_to_buy","acc_open_past_24mths","loan_status"]
X_train, X_test = X_train[features[:-1]], X_test[features[:-1]]
data_clean = data_clean[features]
print(X_train.shape)
print(data_clean.shape)
```

建立分類模型

使用監督式學習進行預測

- 分類問題
  - 根據已知標籤的訓練資料集(Training Set)，產生一個新模型，用以預測測試資料集(Testing Set)的標籤。
  - e.g. 客戶流失分析

- 回歸分析
  - 使用一組**已知對應值**的資料產生的模型，預測新資料的對應值
  - e.g. 股價預測

用機率與規則產生預測模型

■ 規則模型

e.g.

假使使用者為女性

而且月收入高達3萬以上

而且還沒看過這廣告

點擊機率為11%

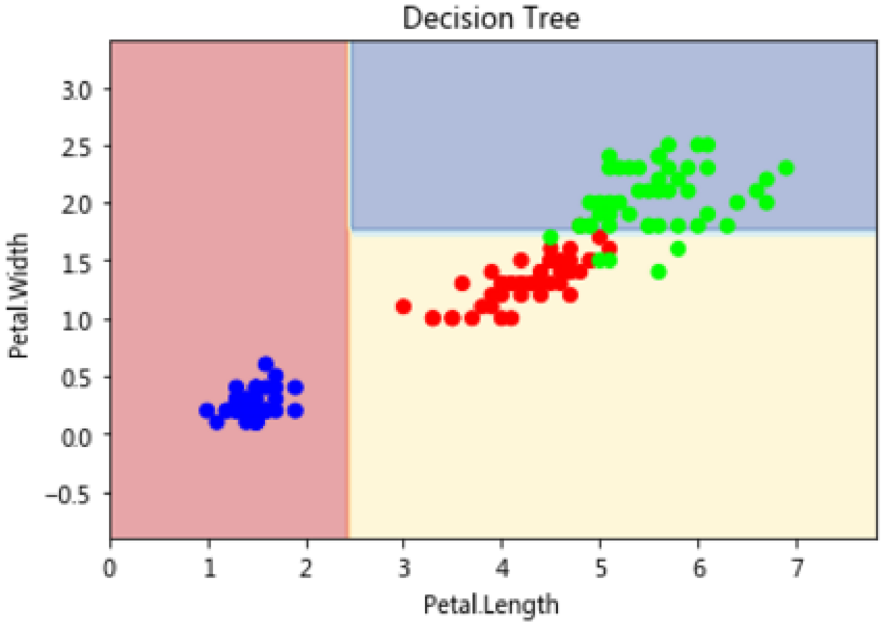
■ 線性模型

e.g. 針對一個化妝品廣告，對女性的吸引力可以給予權重90%，男性權重只有10%，以權重搭配個人點擊機率 (15%)可以算出對該使用者推薦的分數(或機率)

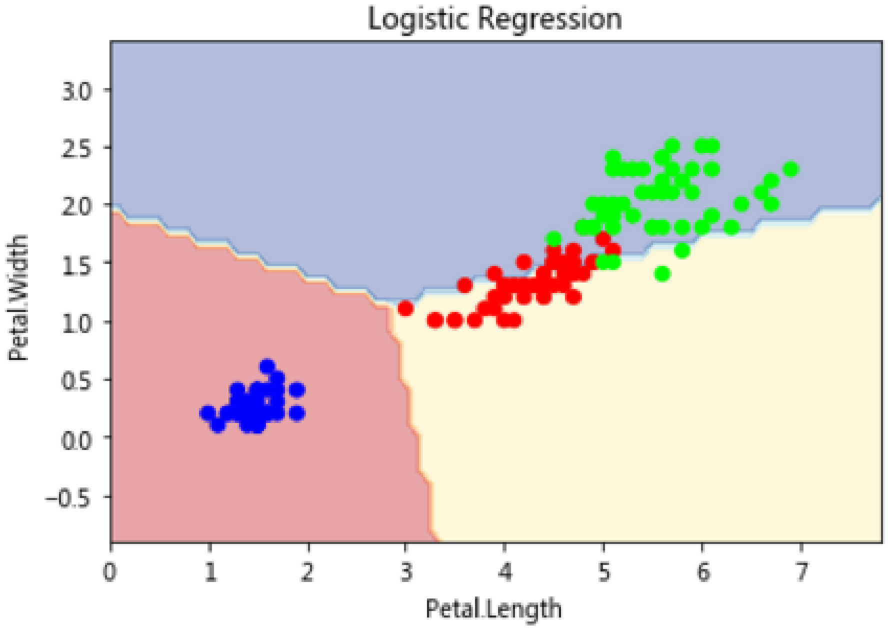
女性13.5%，男性1.5%

模型比較

決策樹- 規則模型



邏輯回歸模型-線性模型

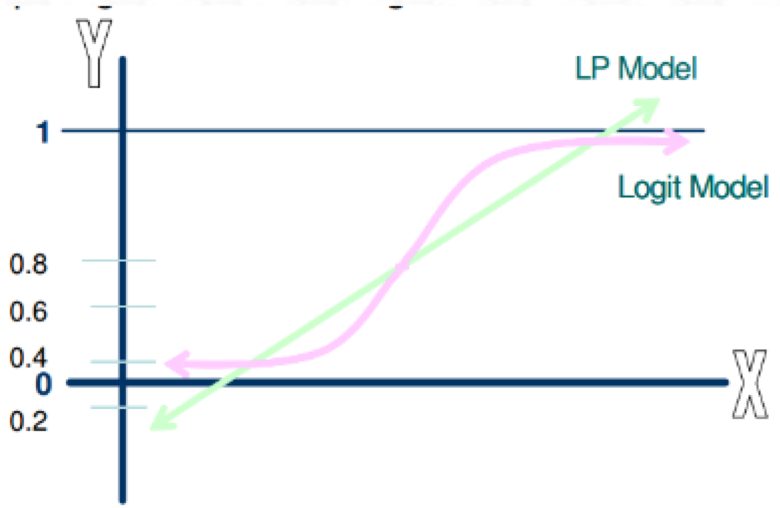


將模型配適到資料資料

- 參數學習(Parameter Learning)
  - 或稱為「參數化建模」(Parametric Modeling)
  - 以未確定的數值參數指定模型結構，依特定的訓練資料算出最佳參數值
  - 先根據專業知識挑選屬性，利用演算法調整參數，讓模型盡可能符合資料

邏輯回歸分析 (Logistic Regression)

- 從對連續依變數的預測轉變為二元的結果(是/否)
  - 客戶是否流失?
  - 客戶是否買單?
  - 腫瘤為良性還惡性?



如果是線性回歸
會不會X值越大會得到
>100% 的預測結果?

邏輯回歸分析 (Logistic Regression)

- 定義

Logistic Regression

$$\log it(y) = \ln(odds) = b_0 + b_1 X_1 + \varepsilon$$

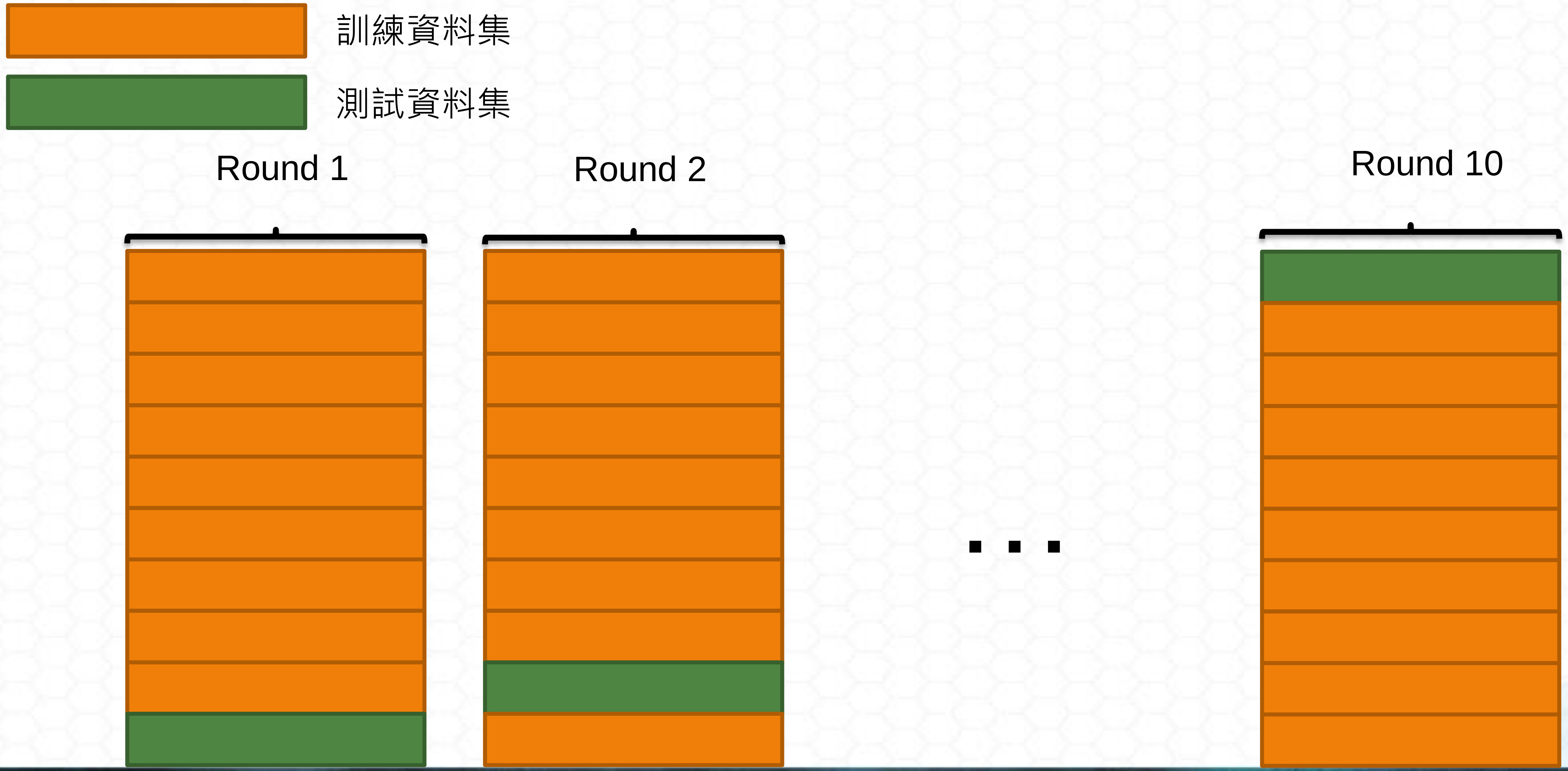
- Odds
  - Odds
    = Probability of event for success (PE)/ failure
    = PE/(1-PE)
- 推導

$$e^{\ln(odds)} = odds = e^{(b_0 + b_1 X_1 + \varepsilon_i)}$$

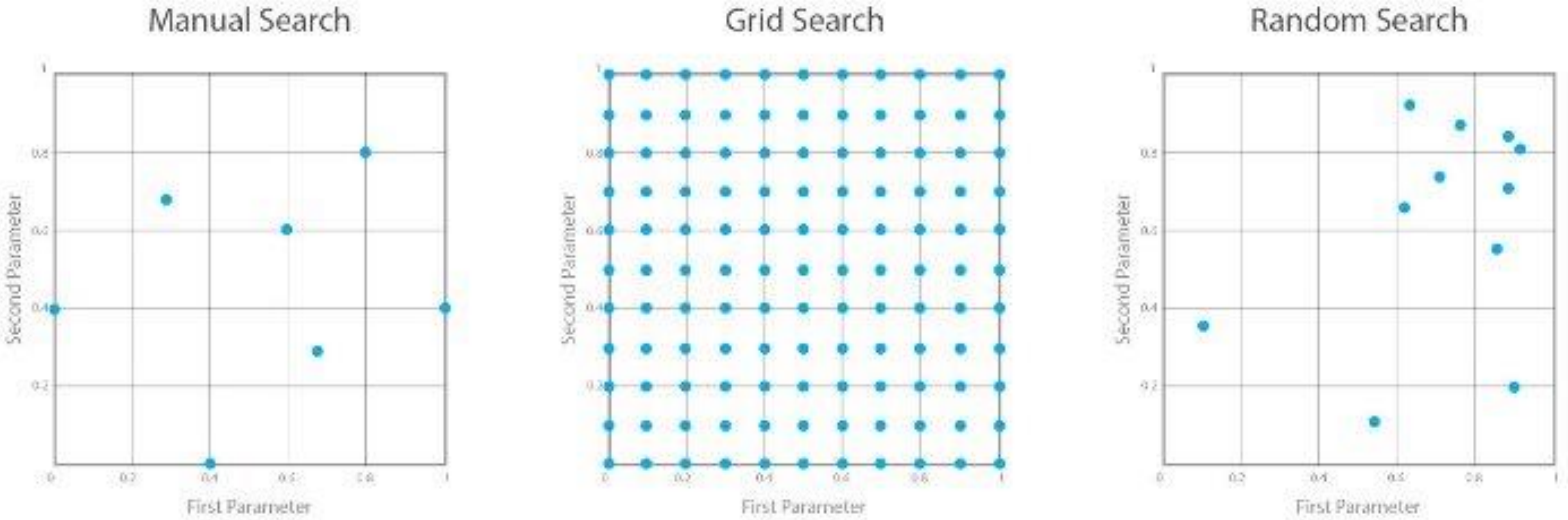
$$PE = odds/(1+Odds) = e^{(b_0 + b_1 X_1 + \varepsilon_i)} * \frac{1}{1 + e^{(b_0 + b_1 X_1 + \varepsilon_i)}}$$

單純代表獲勝/失敗的機率

K-fold 交叉驗證



搜尋超參數


Manual Search
Second Parameter
First Parameter
Grid Search
Second Parameter
First Parameter
Random Search
Second Parameter
First Parameter

Grid Search

```
from sklearn.model_selection import GridSearchCV

def cross_validation_best_parameters(model, param_grid):
    grid = GridSearchCV(model, param_grid,cv=10, scoring='accuracy')
    X=data_clean.iloc[:,:-1].values
    y=data_clean.iloc[:,-1].values
    grid.fit(X,y)
    mean_scores = [result.mean_validation_score for result in grid.grid_scores_]
    return mean_scores,grid.best_score_,grid.best_estimator_
```

搜尋最佳模型

```
logreg = linear_model.LogisticRegression(random_state=0)
c=[0.001, 0.01, 0.1, 1, 10, 100, 1000]
param_grid = dict(C=c)
mean_scores,Best_Accuracy, Best_classifier =
cross_validation_best_parameters(logreg,param_grid)
print("Best accuracy is "+ str(Best_Accuracy))
print(Best_classifier)
```

準確率毫無意義

- 類別資料不平衡的情況下:
  - 假使客戶有1,000人，今天流失的客戶數量是50人，今天假使有一個預測模型的預測準確率有90%，試問這是個好的分類模型嗎?

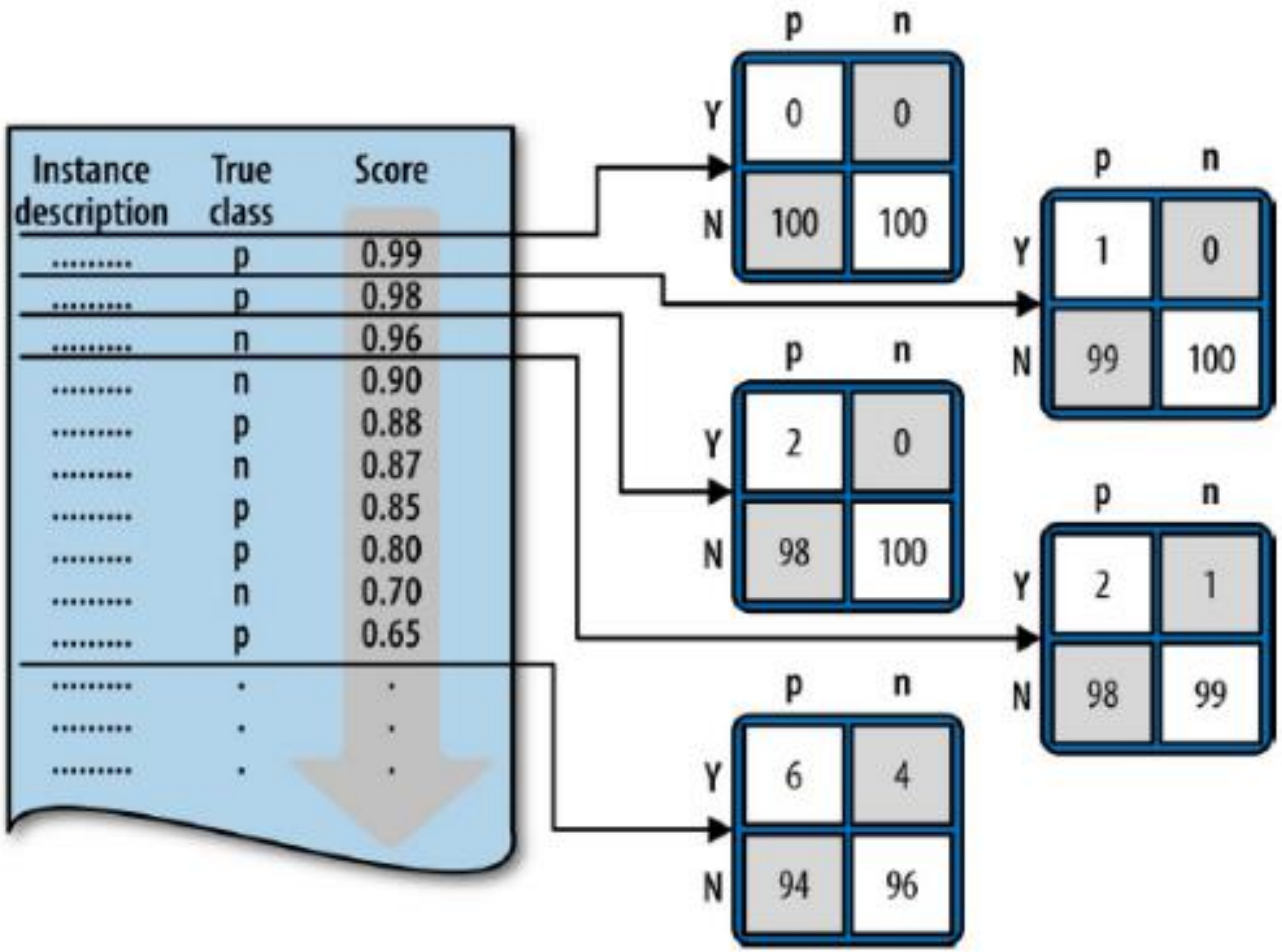
數據如何被分類？如何被分錯？

- 需要有方法可分解並計算由分類器產生不同類型的正誤數量
  - 需要使用混淆矩陣(Confusion Matrix)

混淆矩陣（Confusion Matrix）

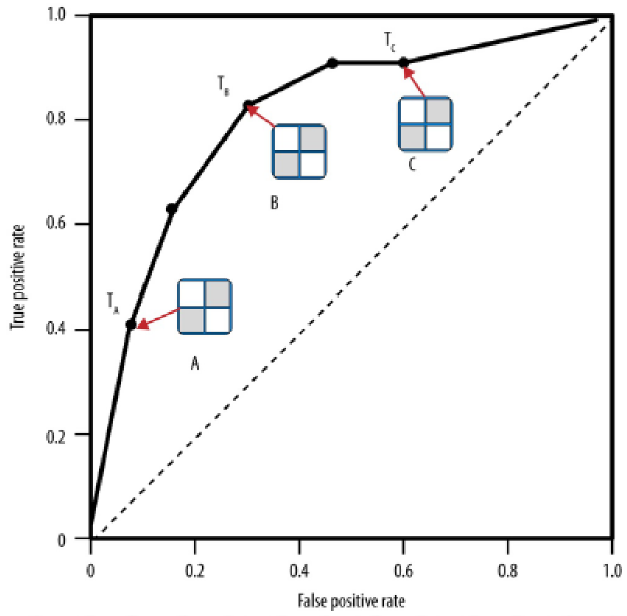
	真	假
有	True Positive	False Positive Type I Error
無	False Negative Type II Error	True Negative

考慮不同成本下的混淆矩陣

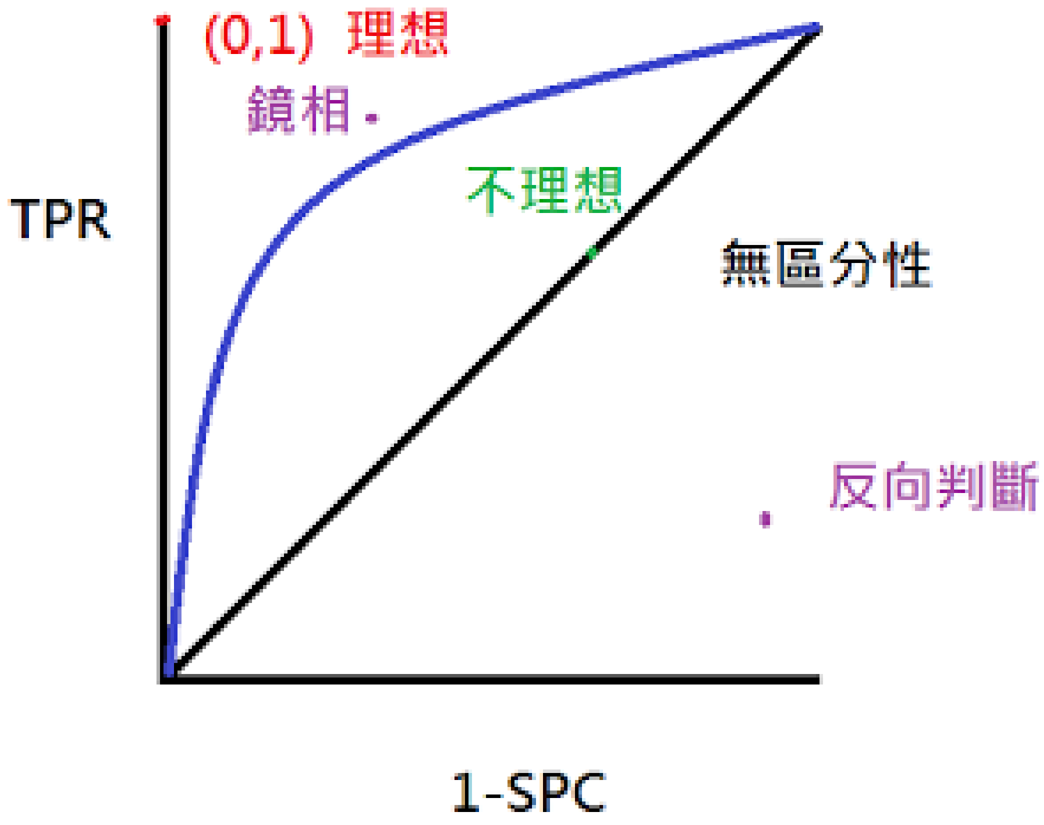


ROC 曲線

- 接收者操作特徵(receiver operating characteristic, ROC curve)
  1.以假陽性率(False Positive Rate, FPR)為X軸，代表在所有陰性相本中，被判斷為陽性(假陽性)的機率，又寫為(1-特異性)。
  2.以真陽性率(True Positive Rate, TPR)為Y軸，代表在所有陽性樣本中，被判斷為陽性(真陽性)的機率，又稱為敏感性



評估 ROC 曲線



AUC

曲線下面積(Area Under Curve, AUC)為此篩檢方式性能優劣之指標，AUC越接近1，代表此篩檢方式效能越佳。指標可參考以下條件。

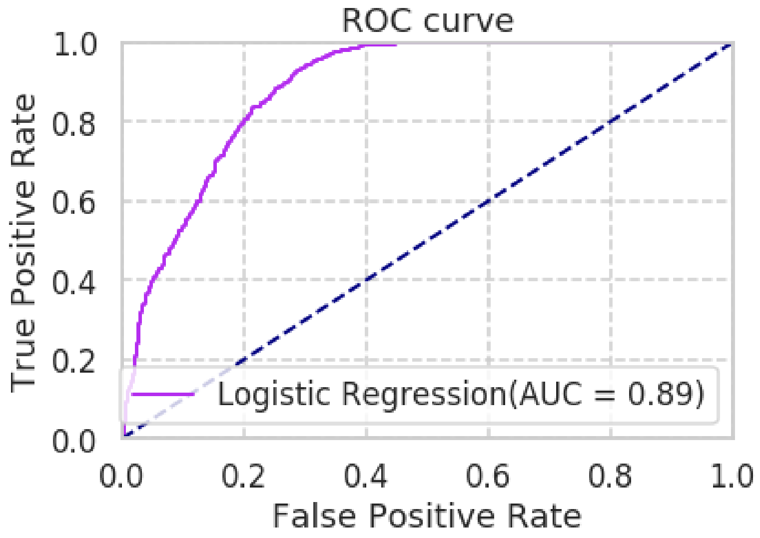
AUC數值	解釋
1	完美分類器，無論cut-off point如何設定都可正確預測。通常不存在
0.5<AUC<1	優於隨機，妥善設定可有預測價值
0.5	同隨機，預測訊息沒有價值

繪製ROC Curve

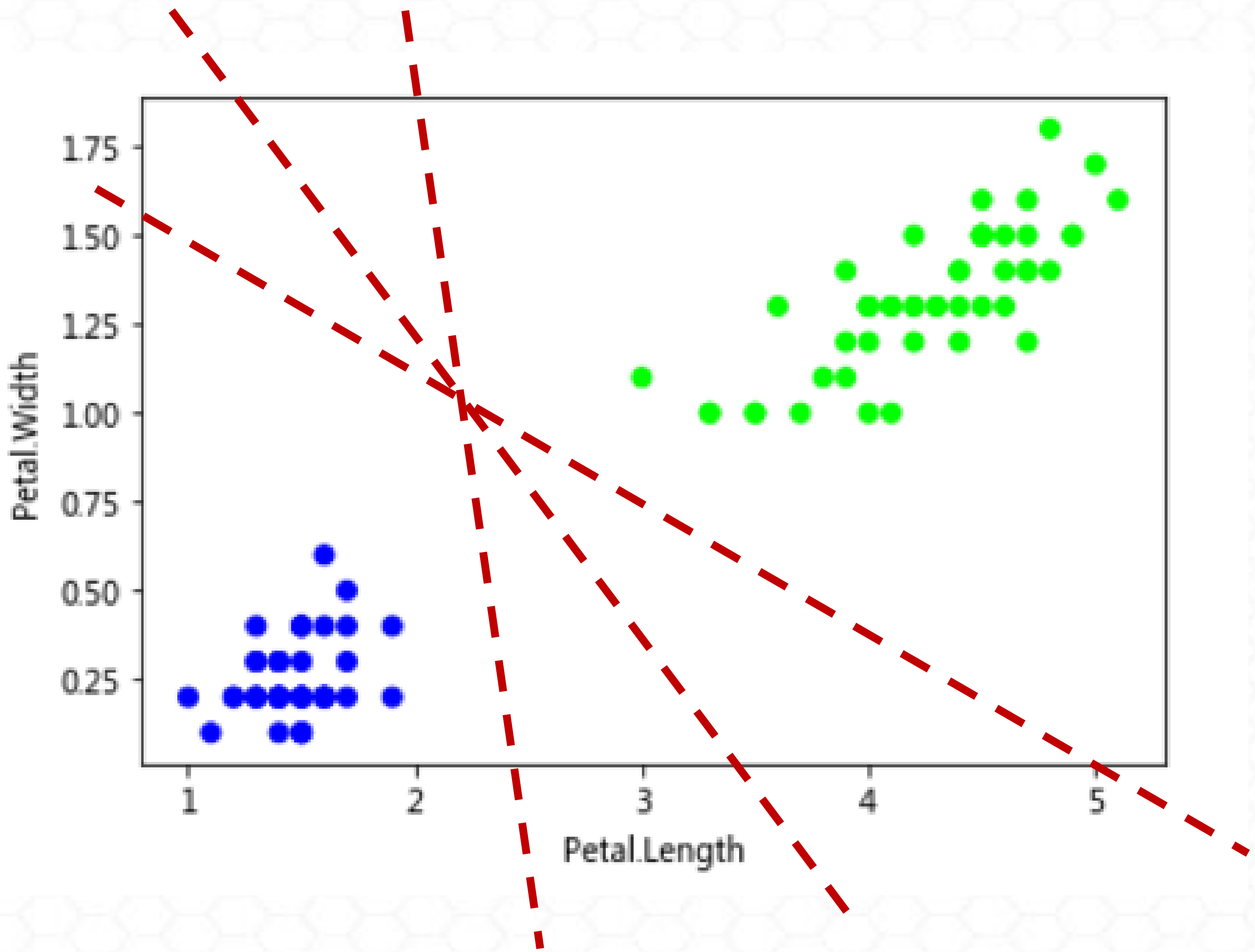
```
import seaborn as sns
sns.set('talk', 'whitegrid', 'dark', font_scale=1, font='Ricty',rc={"lines.linewidth": 2, 'grid.linestyle': '--'})
def plotAUC(truth, pred, lab):
    fpr, tpr, _ = metrics.roc_curve(truth,pred)
    roc_auc = metrics.auc(fpr, tpr)
    lw = 2
    c = (np.random.rand(), np.random.rand(), np.random.rand())
    plt.plot(fpr, tpr, color= c,lw=lw, label= lab +'(AUC = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.0])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC curve') #Receiver Operating Characteristic
    plt.legend(loc="lower right")
```

繪製ROC Curve

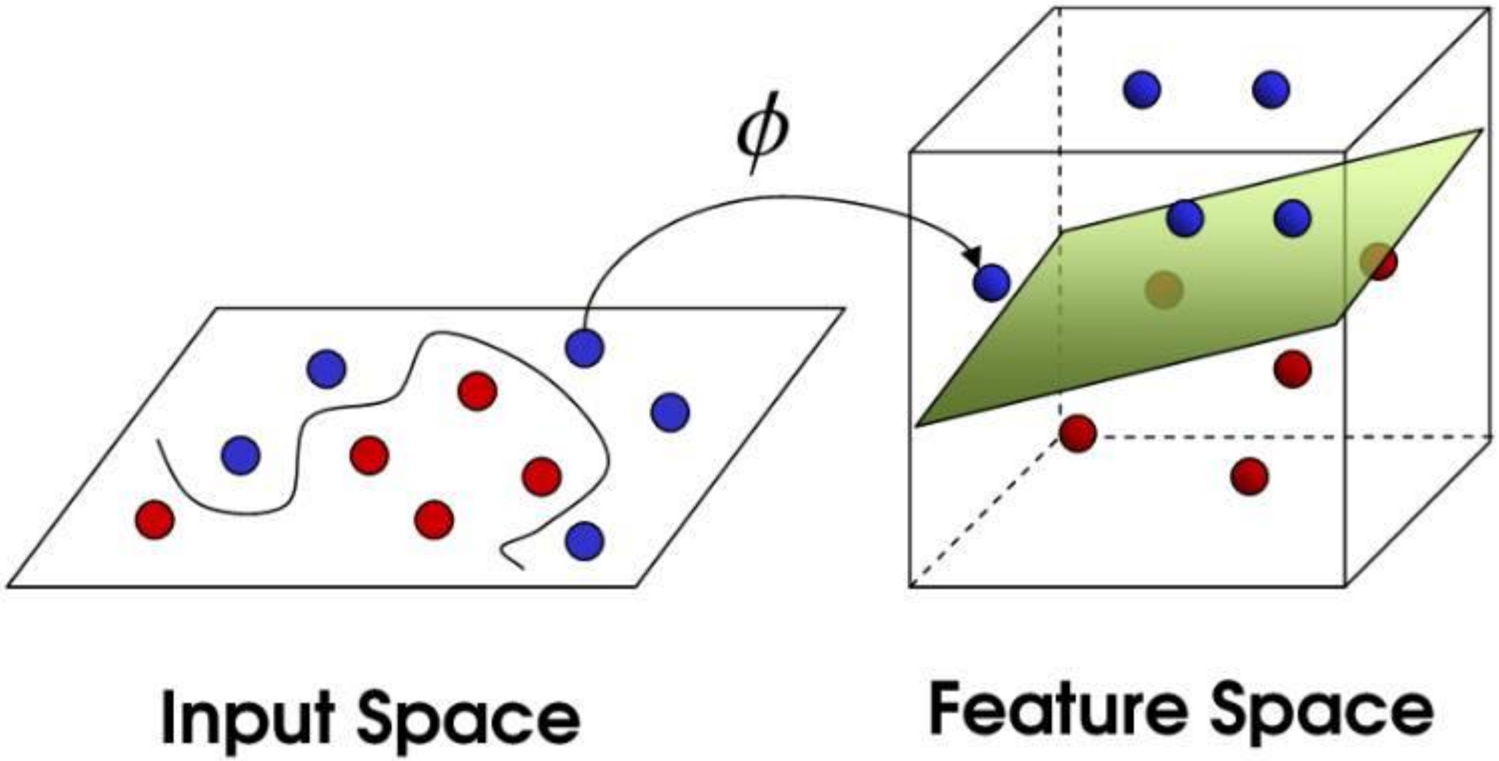
```
clf_LR = linear_model.LogisticRegression(C=Best_classifier.C)
clf_LR.fit(X_train,y_train)
LR_Predict = clf_LR.predict_proba(X_test)[:,1]
LR_Predict_bin = clf_LR.predict(X_test)
LR_Accuracy = accuracy_score(y_test,LR_Predict.round())
print("Logistic regression accuracy is ",LR_Accuracy)
plotAUC(y_test,LR_Predict,'Logistic Regression')
plt.show()
```



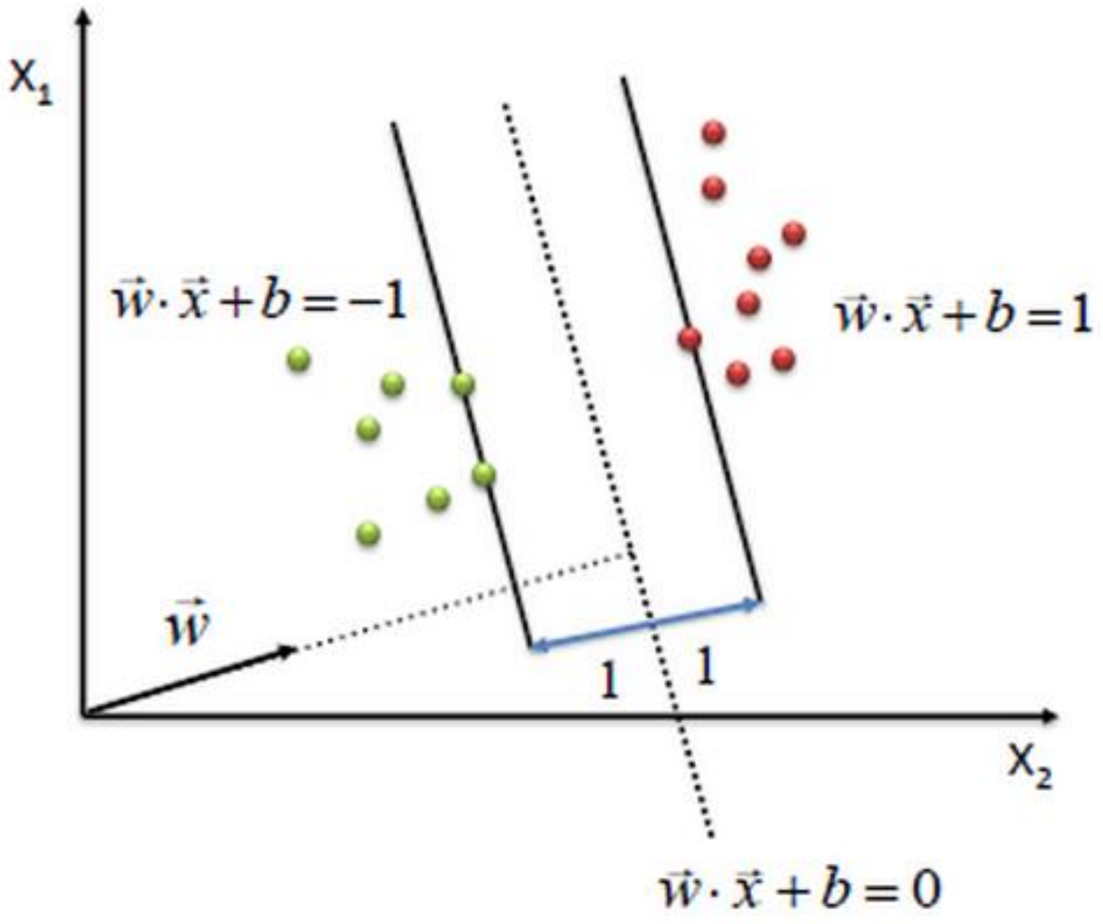
該選哪一條線做切割?



如何解決高維度資料切分問題



支持向量機 (Support Vector Machine)



$$\boxed{\max \frac{2}{\|w\|}}$$

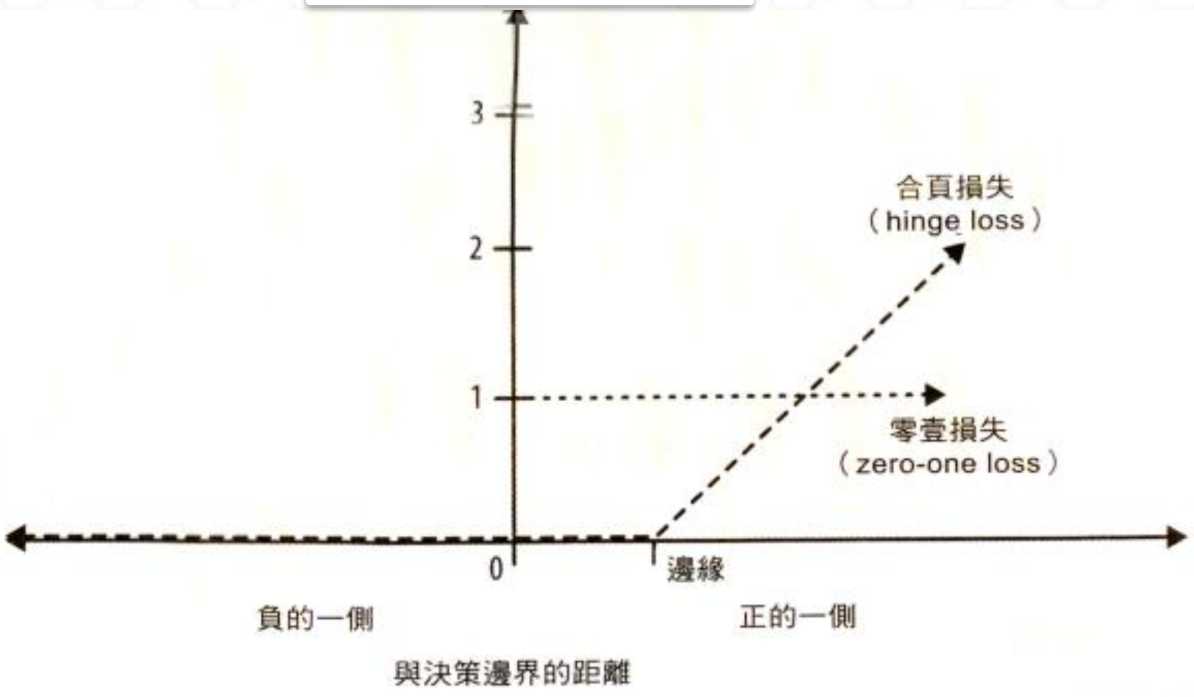
$s.t.$

$(w\cdot x+b)\geq 1, \forall x \text{ of class } 1$

$(w\cdot x+b)\leq -1, \forall x \text{ of class } 2$

評估決策邊界

SVM 只會犯小錯



SVM

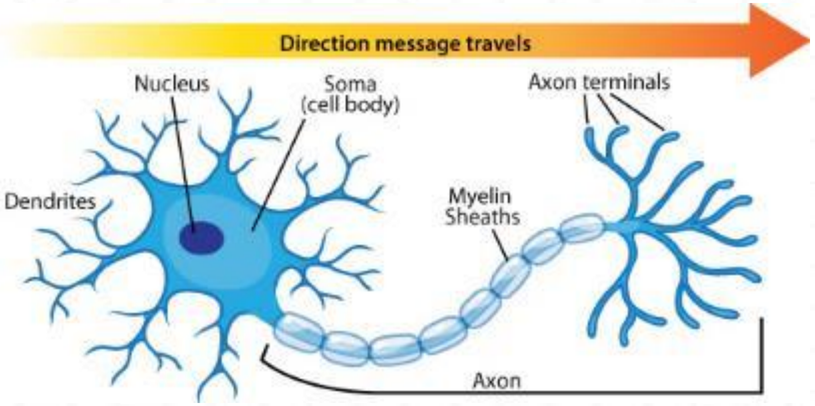
```
from sklearn.grid_search import GridSearchCV
clf_svm = svm.SVC()
powers = range(0,5)
cs = [10**i for i in powers]
param_grid = dict(C=cs)
grid = GridSearchCV(clf_svm, param_grid, cv=10, scoring='accuracy')
grid.fit(data_clean.iloc[:,:-1].values, data_clean.iloc[:,-1].values)
grid_mean_scores = [result.mean_validation_score for result in grid.grid_scores_]
print("---------------")
print(grid.best_estimator_)
```

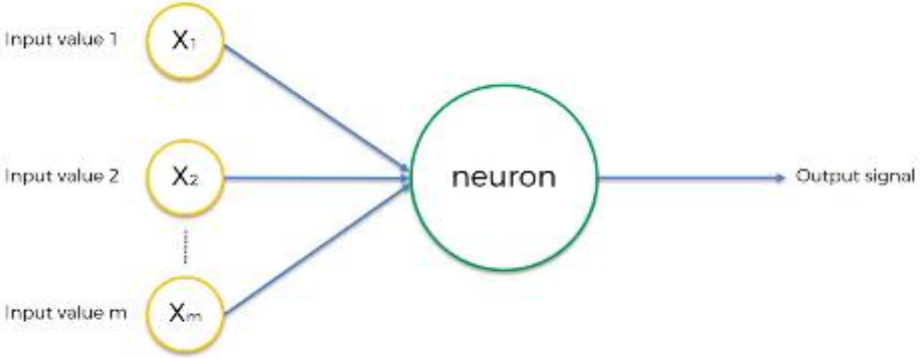
評估SVM模型

```
clf_svm = svm.SVC(kernel = "rbf", C=grid.best_estimator_.C)
clf_svm.fit(X_train.iloc[:,:],y_train)
predictions_svm = clf_svm.predict(X_test.iloc[:,:])
predictproba_svm = clf_svm.decision_function(X_test.iloc[:,:])
SVM_Accuracy = accuracy_score(y_test,predictions_svm)
print("SVM accuracy is ",SVM_Accuracy)
plotAUC(y_test,predictproba_svm, 'SVM')
plotAUC(y_test,LR_Predict,'Logistic Regression')
plt.show()
```

類神經網路

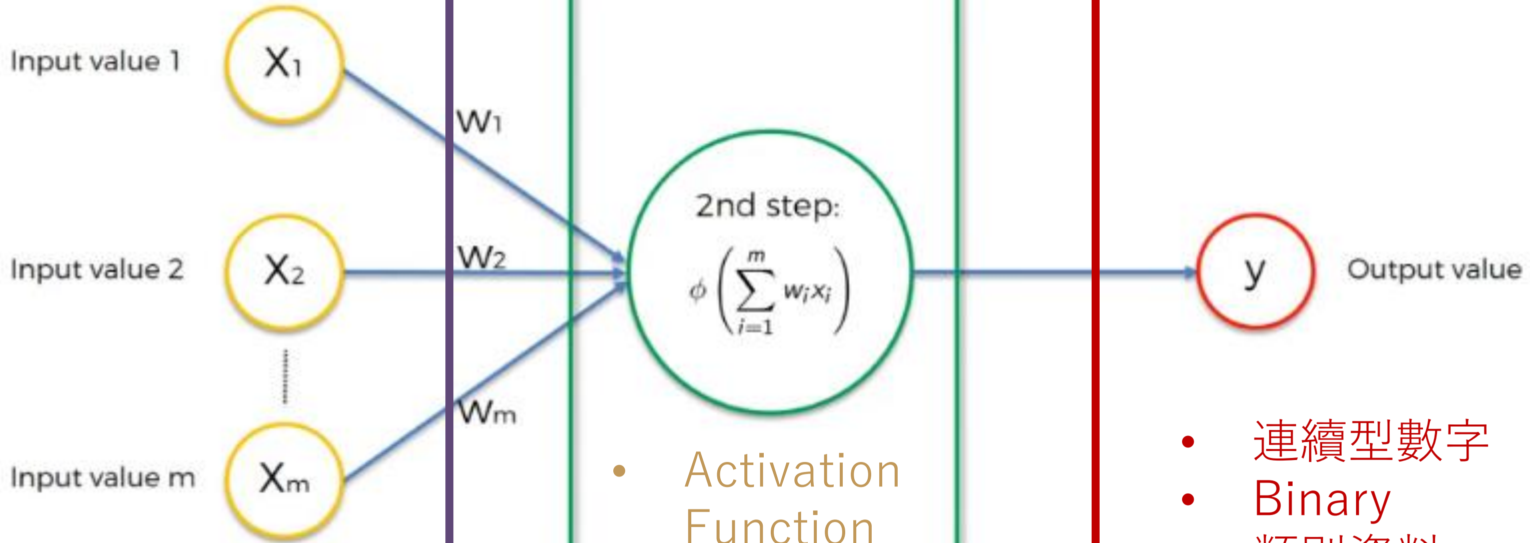
類神經網路





1. 加總收集到的訊號
2. 非線性轉換
3. 產生一個新的信號

神經元

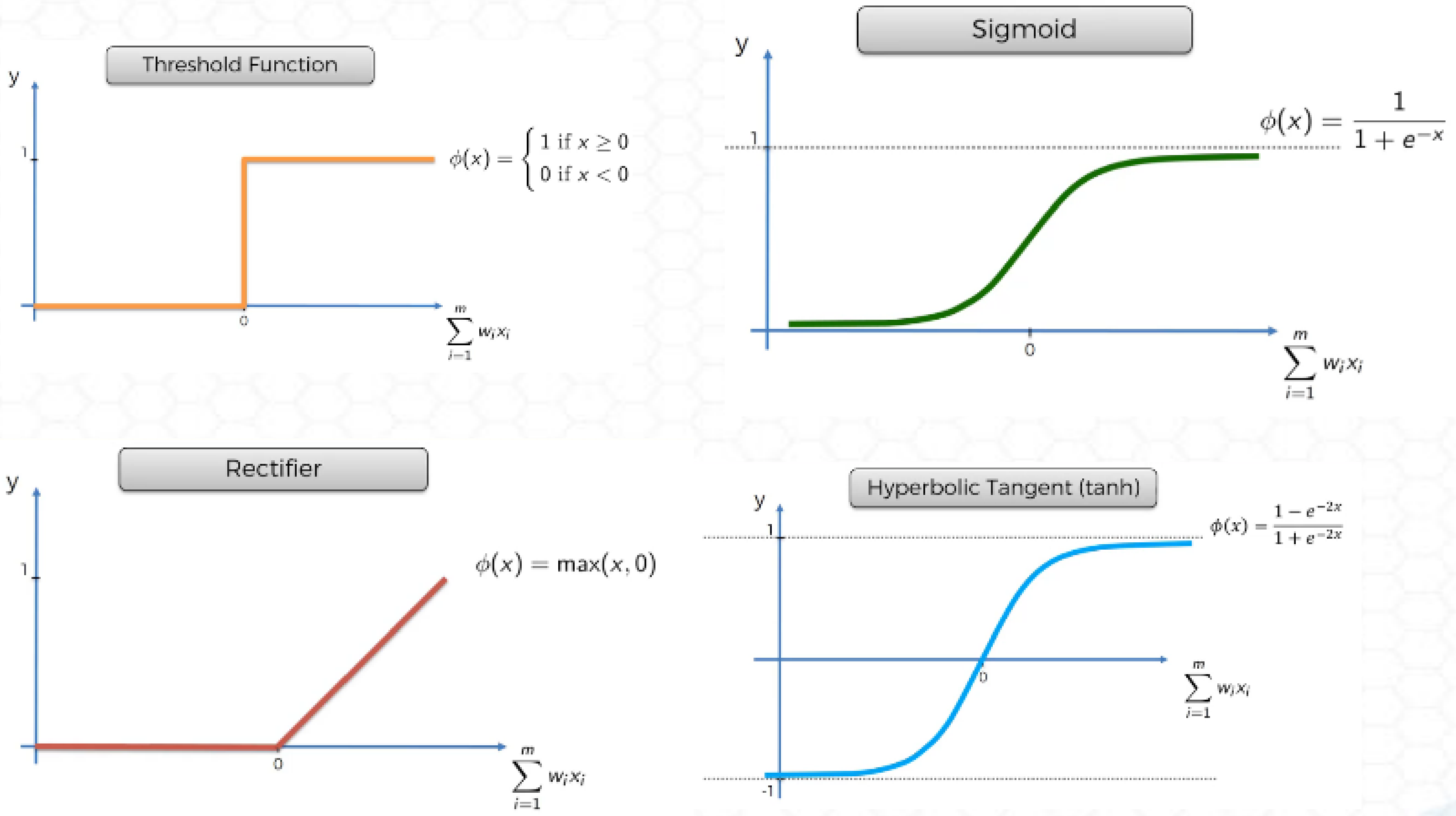


- 來自同一個觀察的獨立變數
- 必須要標準化

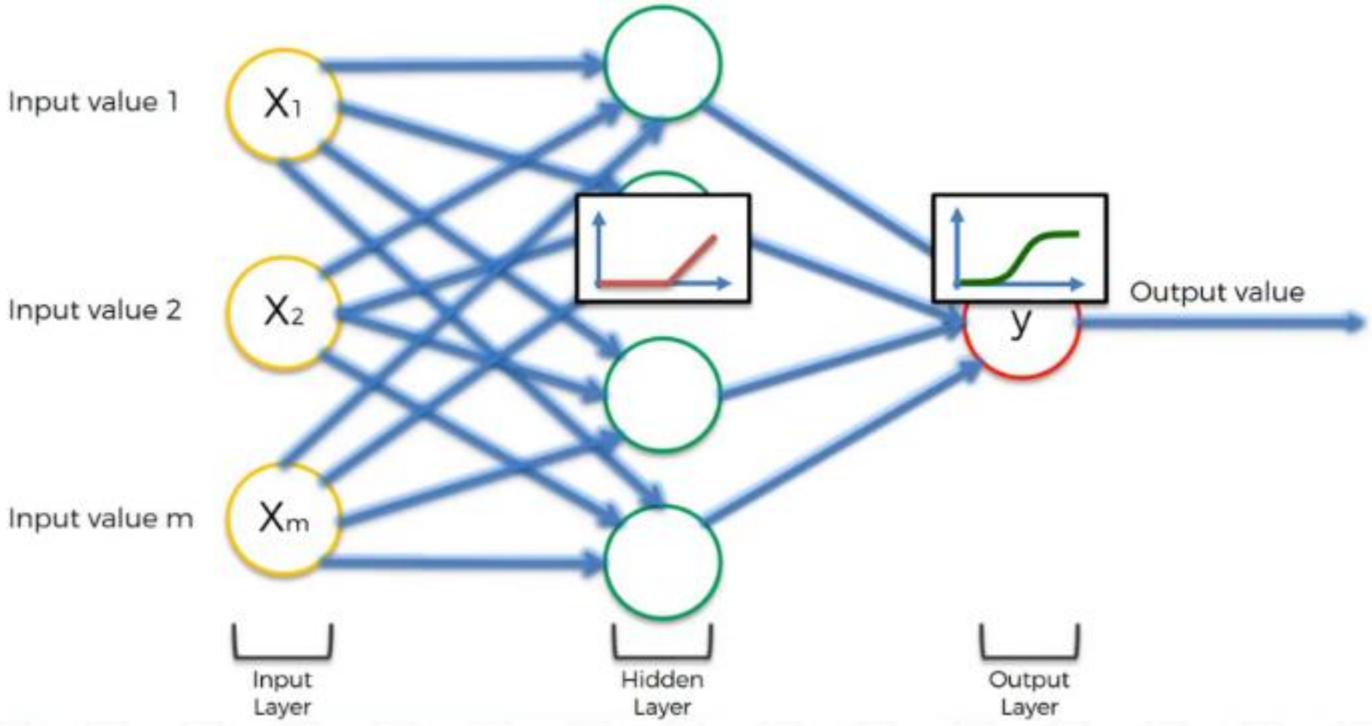
- Activation Function

- 連續型數字
- Binary
- 類別資料

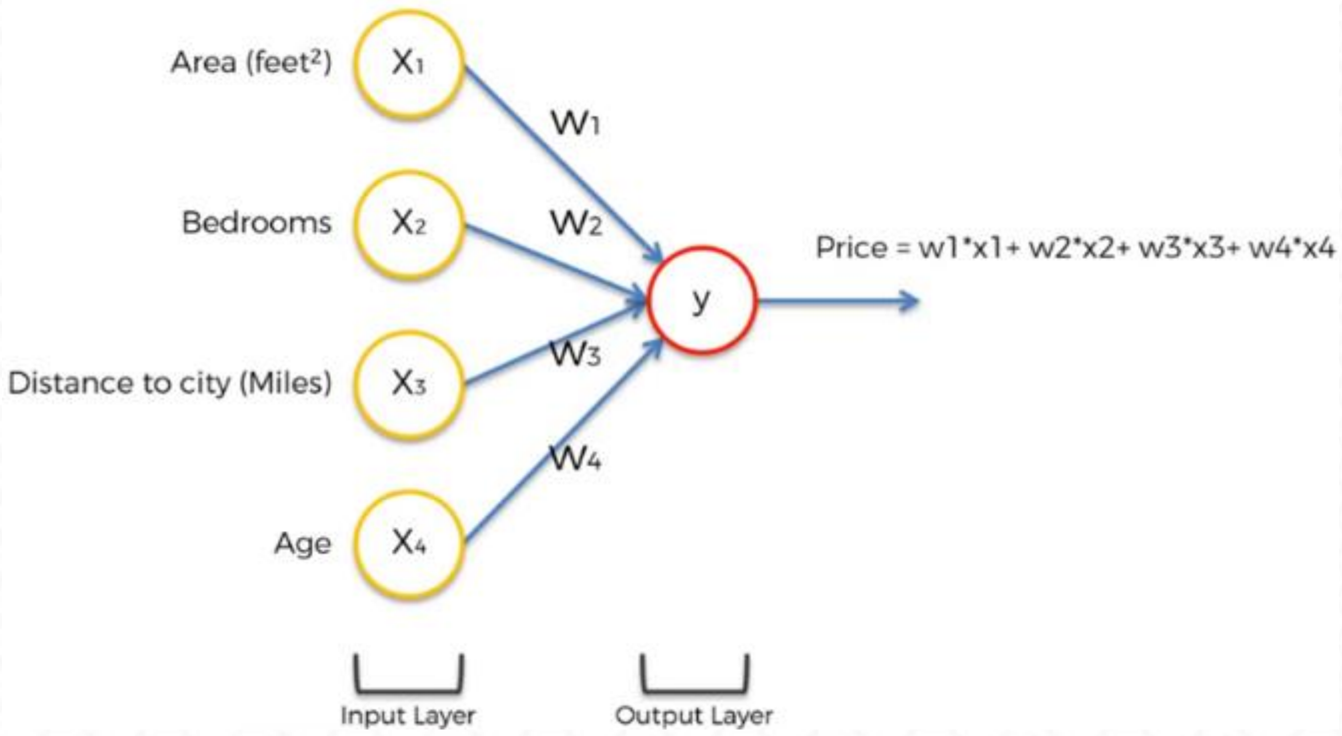
激勵函數(Activation Function)



激勵函數(Activation Function)

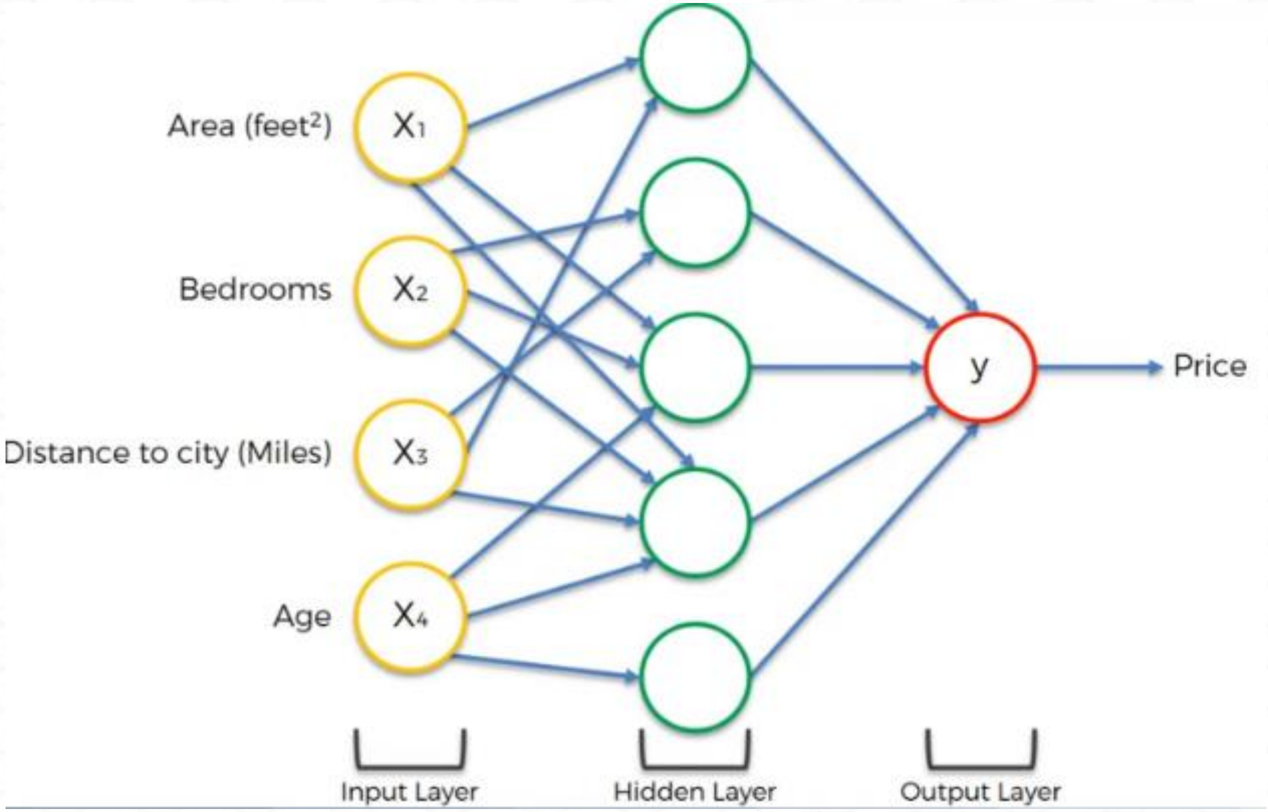


兩層神經網路



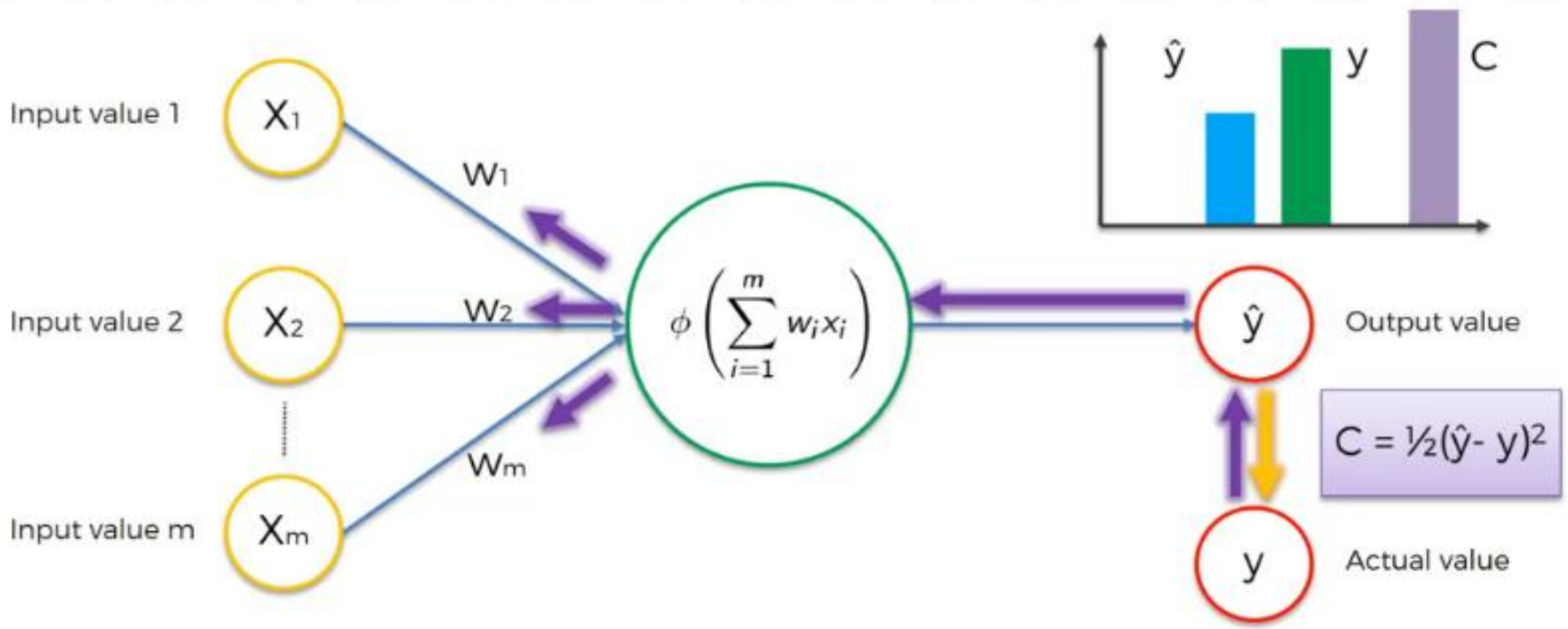
Area (feet²)
Bedrooms
Distance to city (Miles)
Age
X1
X2
X3
X4
W1
W2
W3
W4
y
Price = w1*x1+ w2*x2+ w3*x3+ w4*x4
Input Layer
Output Layer

三層神經網路



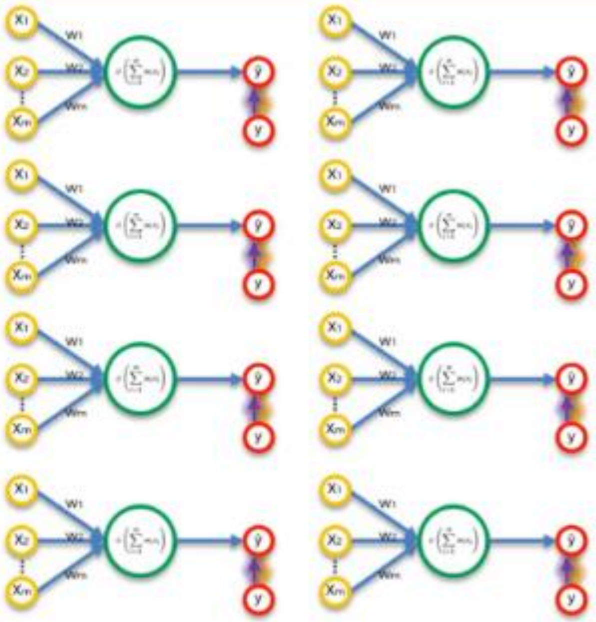
Area (feet²)
X1
Bedrooms
X2
Distance to city (Miles)
X3
Age
X4
y
Price
Input Layer
Hidden Layer
Output Layer

類神經網路如何運作



Input value 1
Input value 2
Input value m
X1
X2
Xm
W1
W2
Wm
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
ŷ
Output value
y
Actual value
C = ½(ŷ- y)²
ŷ
y
C

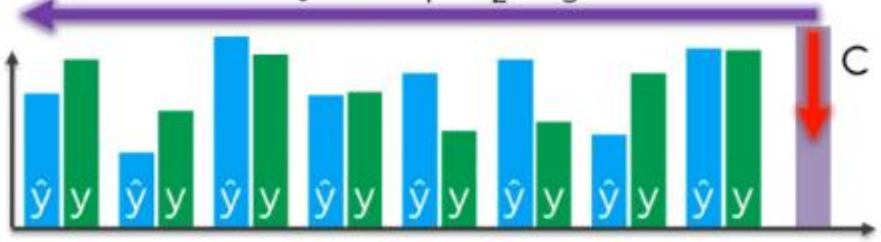
如何根據資料調整權重



Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

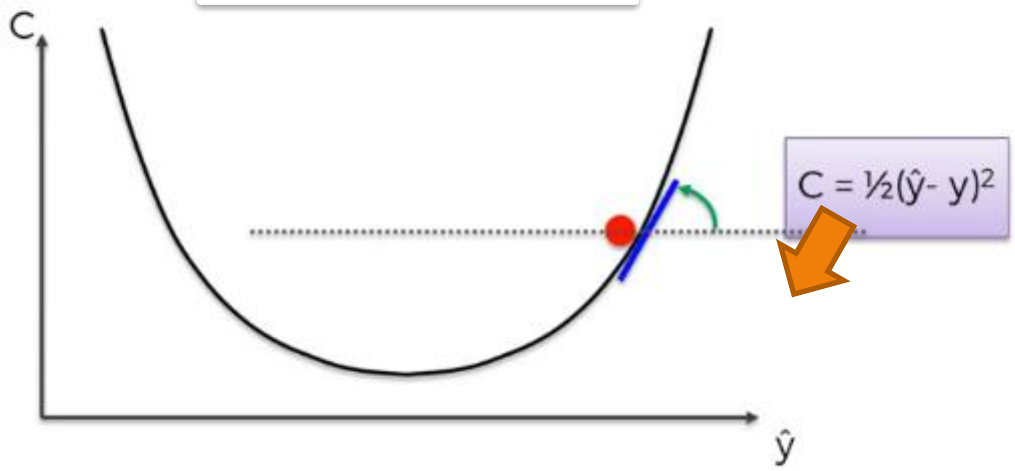
$$C = \sum \tfrac{1}{2}(\hat{y} - y)^2$$

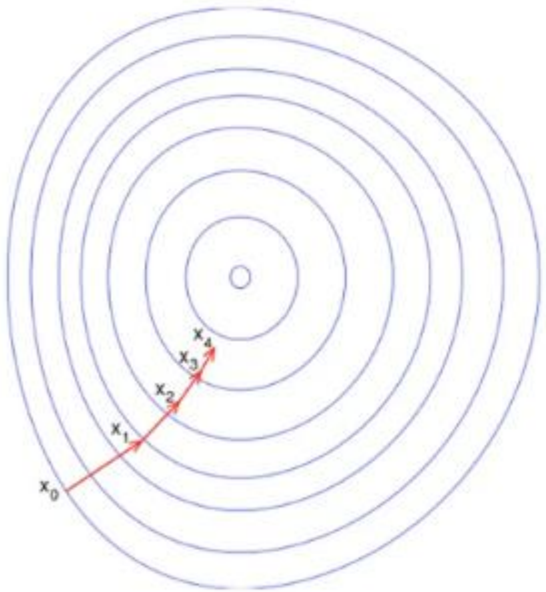
Adjust w_1, w_2, w_3



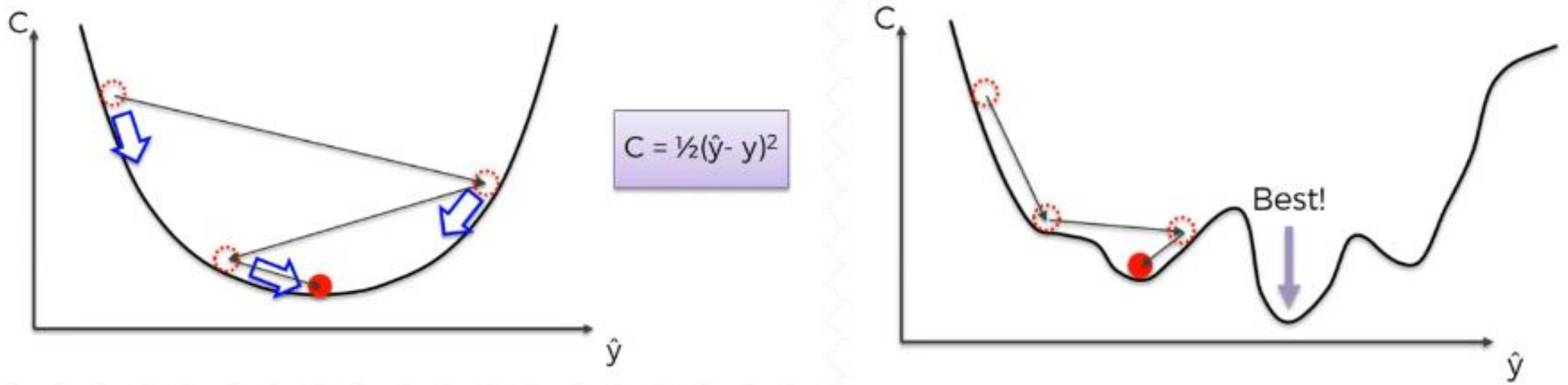
Gradient Descent

透過偏微分方式
找到最佳點





Local Minima



類神經網路

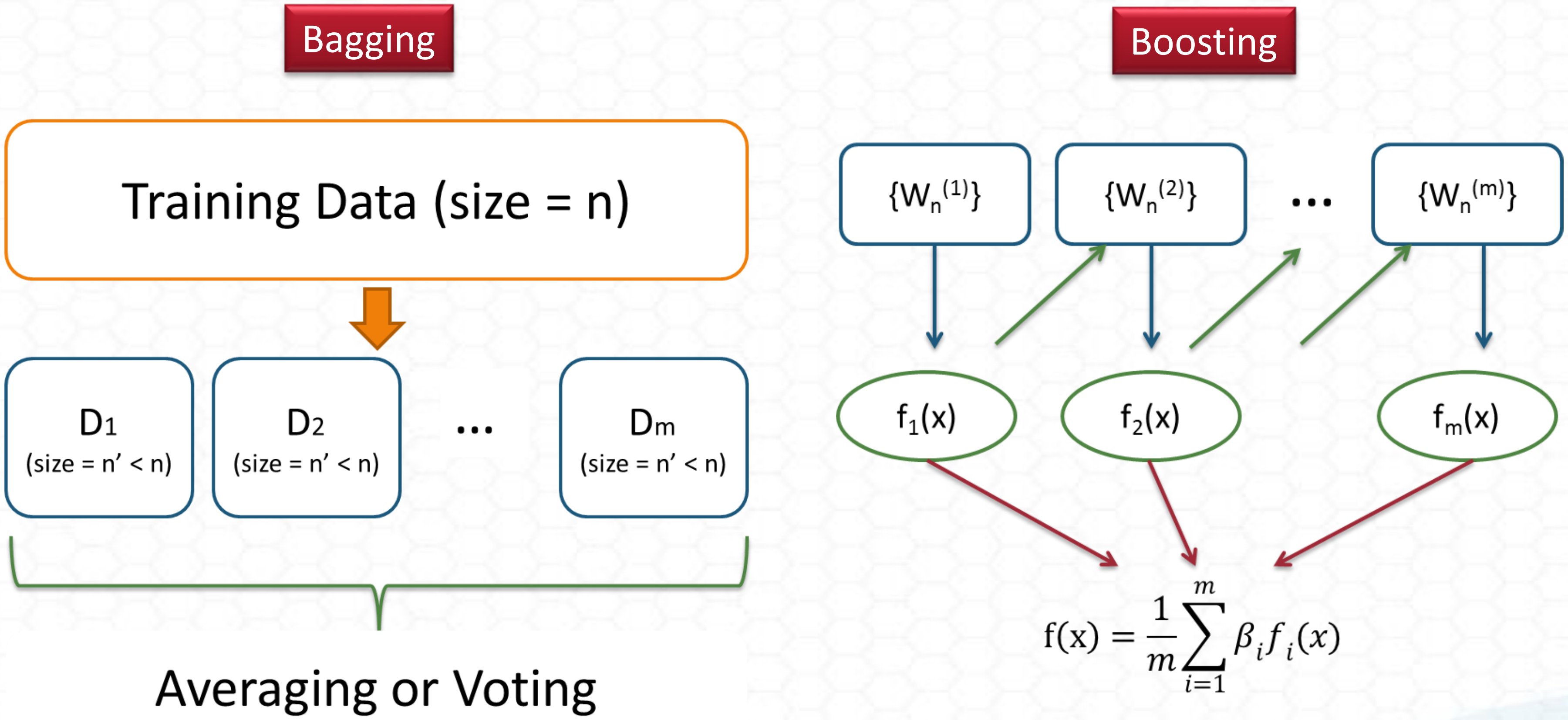
```
from sklearn.neural_network import MLPClassifier
clf_NN = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(5, 2), random_state=1)
clf_NN.fit(X_train,y_train)
predict_NN = clf_NN.predict(X_test)
predictproba_NN = clf_NN.predict_proba(X_test)[:,1]
NNAccuracy = accuracy_score(y_test,predict_NN)
print(NNAccuracy)
```

類神經網路

```
plotAUC(y_test,LR_Predict,'Logistic Regression')
plotAUC(y_test,predictproba_svm, 'SVM')
plotAUC(y_test,predictproba_NN,'MLP')
plt.show()
plt.figure(figsize=(6,6))
plot_confusion_matrix(predict_NN, normalize=True)
plt.show()
```

集成模型

Bagging & Boosting

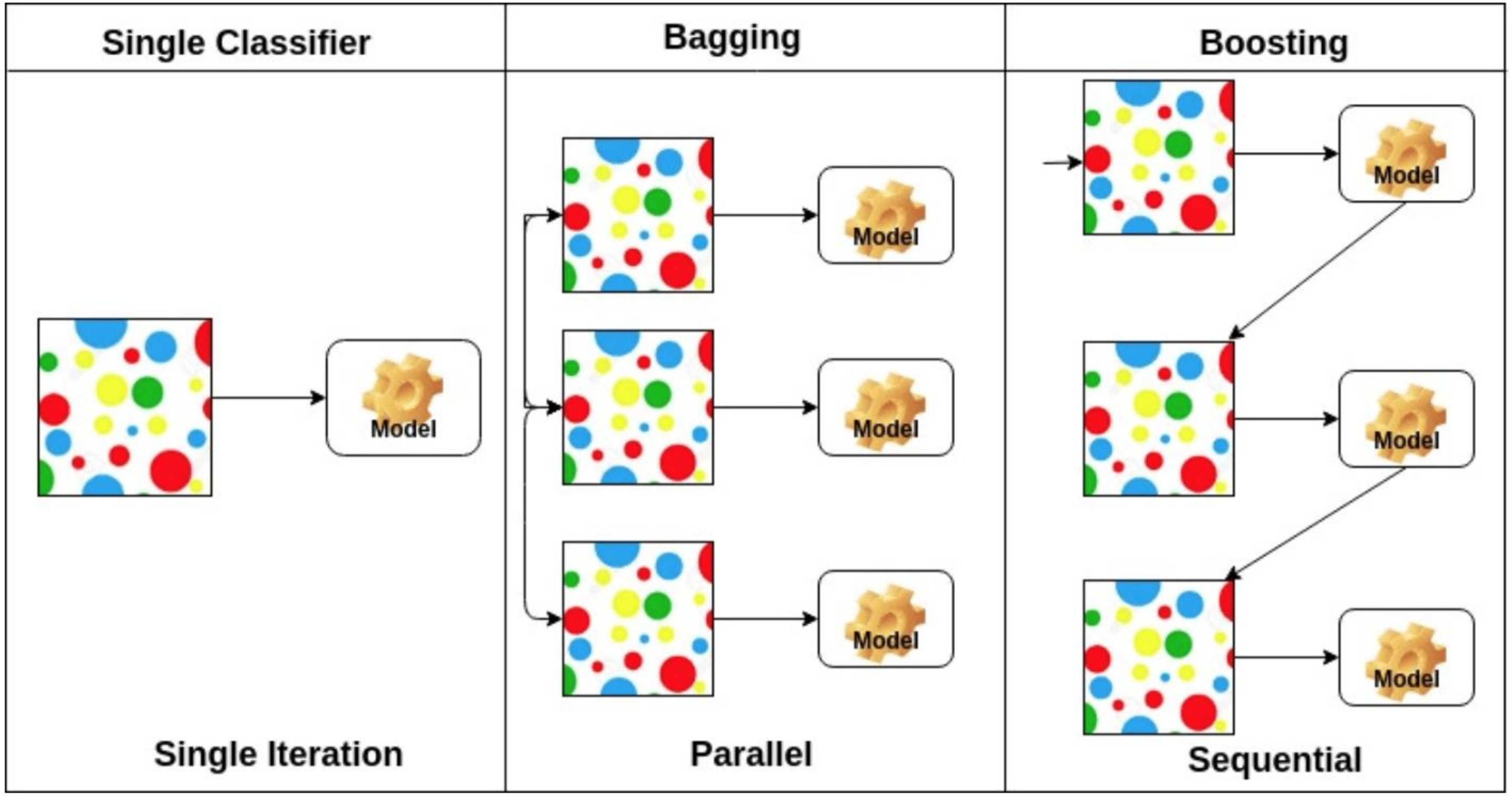


Bagging & Boosting

- **Bagging** (bootstrap aggregation)
  - 合併多個學習器以降低預測的誤差

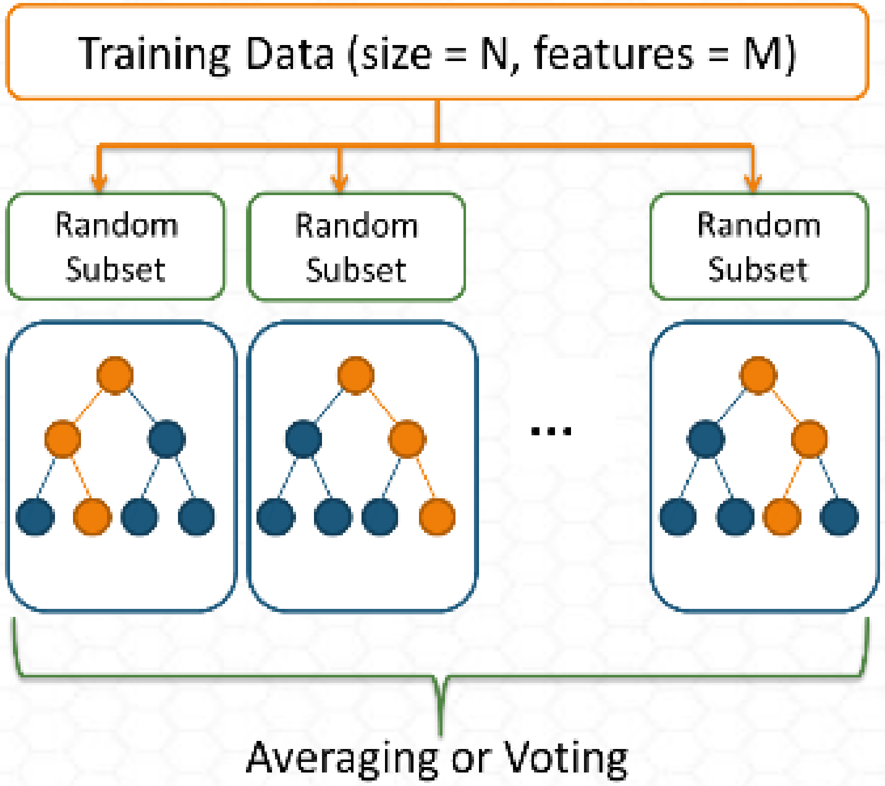
- **Boosting**
  - 利用多個低準度的分類器創建出一個高準度的分類器
  - Boosting 演算法可以找出哪個分類器做出錯誤預測
  - Boosting 可以有效避免過度適配的問題
  - Boosting 演算法
    - AdaBoost (Adaptive Boosting)
    - Gradient Tree Boosting
    - XGBoost

Bagging & Boosting



隨機森林 (Random Forest)

- N 多少樹, M 多少個特徵



使用 Randomized Search 搜尋超參數

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.grid_search import RandomizedSearchCV
rf = RandomForestClassifier(criterion='gini', random_state=0)
maxFeatures = range(1,data_clean.shape[1]-1)
param_dist = dict(max_features=maxFeatures)
rand = RandomizedSearchCV(rf, param_dist, cv=10, scoring='accuracy',
n_iter=len(maxFeatures), random_state=10)
X=data_clean.iloc[:,:-1].values
y=data_clean.iloc[:,-1].values
rand.fit(X,y)
mean_scores = [result.mean_validation_score for result in rand.grid_scores_]
#print('Best Accuracy = '+str(rand.best_score_))
print(rand.best_estimator_)
```

使用最佳參數建立隨機森林模型

```
from sklearn.metrics import accuracy_score
randomForest = RandomForestClassifier(bootstrap=True,criterion = "gini",max_features=rand.best_estimator_.max_features,random_state=0 )
randomForest.fit(X_train,y_train)
```

計算準確率

產生預測結果

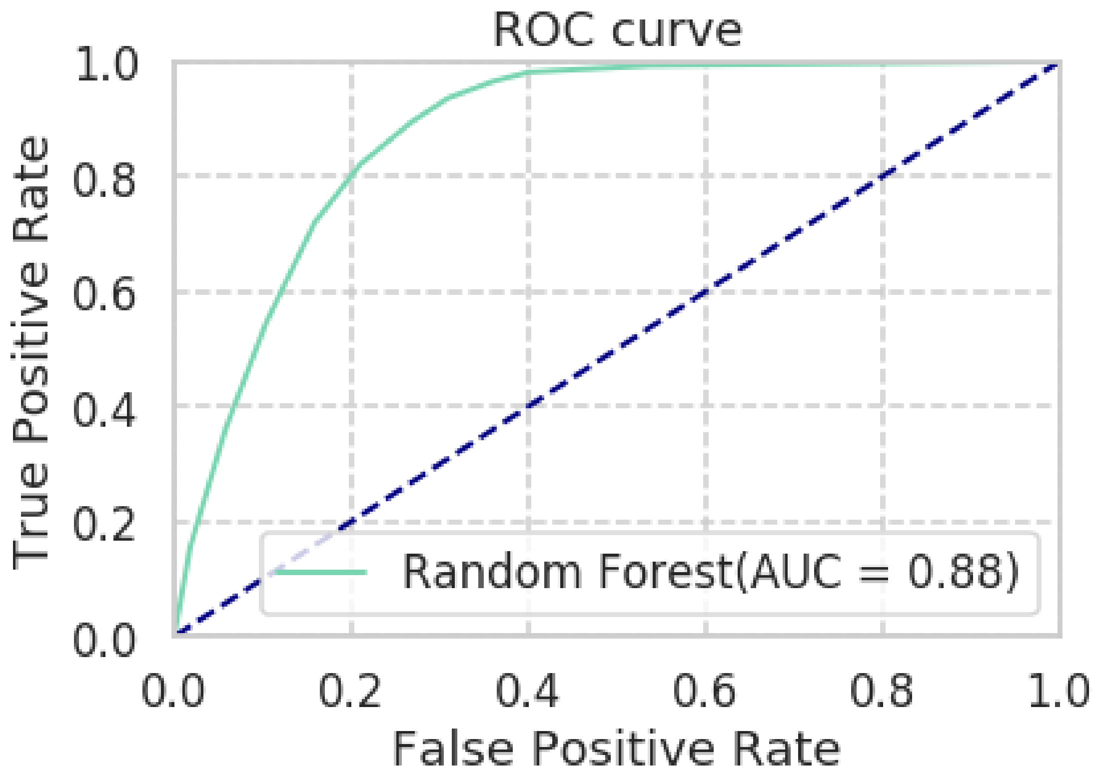
```
rfPredict = randomForest.predict(X_test)
rfAccuracy = accuracy_score(y_test,rfPredict)
print(rfAccuracy)
```

計算 AUC

```
rfPredictproba = randomForest.predict_proba(X_test)[:,1]
roc_score = metrics.roc_auc_score(y_test,rfPredict)
```

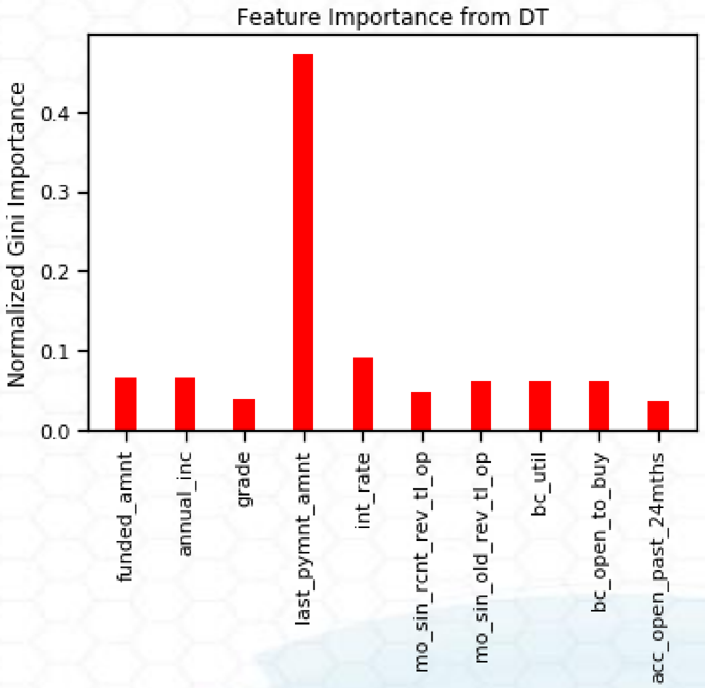
繪製ROC Curve

```
plotAUC(y_test,rfPredictproba, 'Random Forest')
plt.show()
```



特徵重要性

```
fig, ax = plt.subplots()
width=0.35
ax.bar(np.arange(len(features)-1), randomForest.feature_importances_, width, color='r')
ax.set_xticks(np.arange(len(randomForest.feature_importances_)))
ax.set_xticklabels(X_train.columns.values,rotation=90)
plt.title('Feature Importance from DT')
ax.set_ylabel('Normalized Gini Importance')
```



Bagging 法

- 從資料集隨機採樣子樣本
- 根據資料建立分類樹或迴歸樹模型
- 給予新資料，計算每個模型的平均準確度

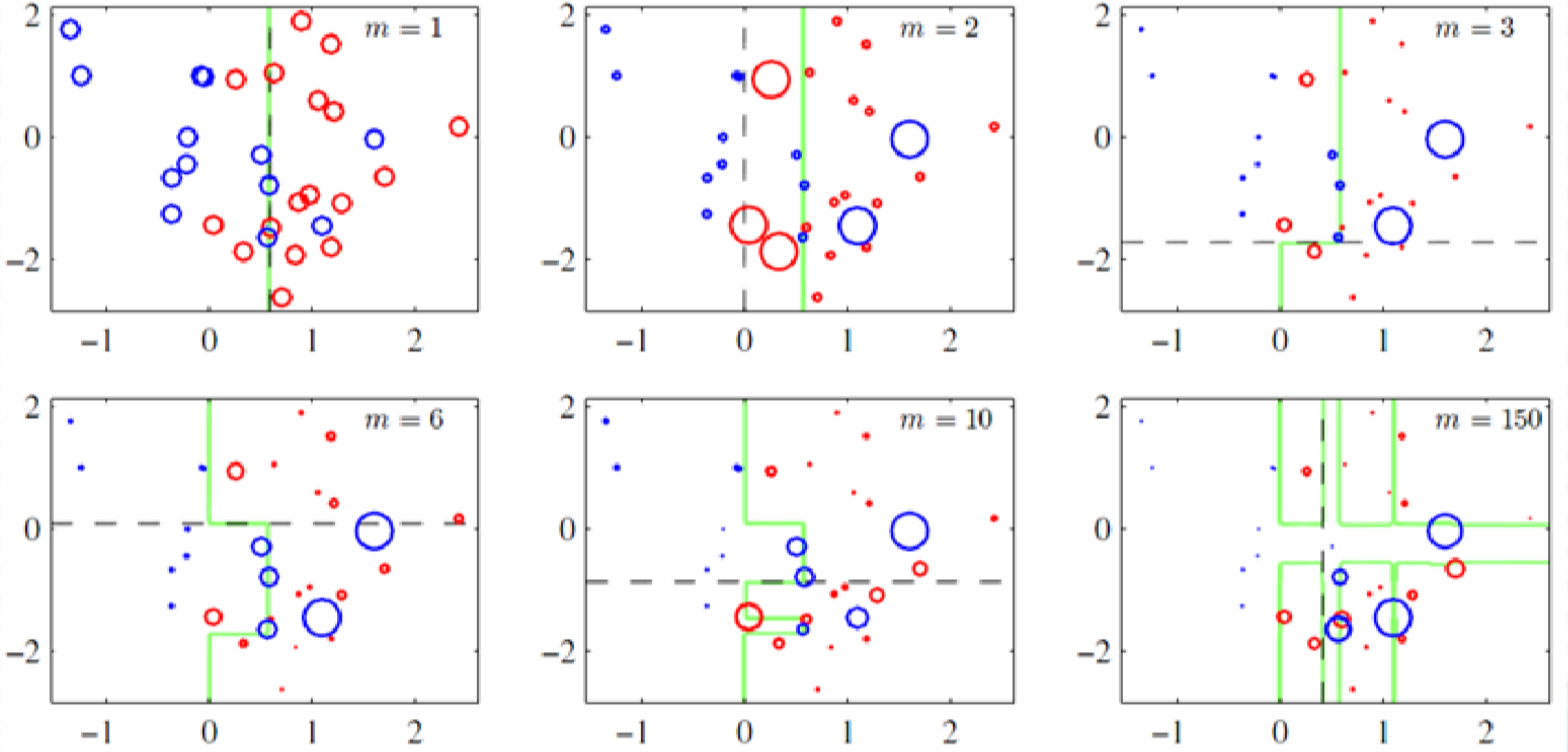
建立Bagging 模型

```
from sklearn import model_selection
from sklearn.ensemble import BaggingClassifier
seed = 7
kfold = model_selection.KFold(n_splits=10, random_state=seed)
num_trees = 100
model = BaggingClassifier(base_estimator=randomForest,
n_estimators=num_trees, random_state=seed)
results = model_selection.cross_val_score(model, data_clean.iloc[:,:-1].values,
data_clean.iloc[:,-1].values, cv=kfold)
print(results.mean())
```

建立Bagging 模型

```
from sklearn import model_selection
from sklearn.ensemble import BaggingClassifier
seed = 7
kfold = model_selection.KFold(n_splits=10, random_state=seed)
#num_trees = 100
model = BaggingClassifier(base_estimator=clf_LR, random_state=seed)
results = model_selection.cross_val_score(model, data_clean.iloc[:,:-1].values,
data_clean.iloc[:,-1].values, cv=kfold)
print(results.mean())
```

Boosting



Boost 演算法

- Boost:
  - 開始時為每個樣本賦予權重值，一開始所有樣本的權重相同
  - 在每一步訓練中得到的模型，會使得數據點的估計有對有錯，在每一步結束後，增加分錯的點的權重，減少分對的點的權重，常被分錯的樣本，會被賦予一個很高的權重
  - 進行了N次運算，將會得到N個簡單的分類器（basic learner），然後我們將它們組合起來，得到一個最終的模型

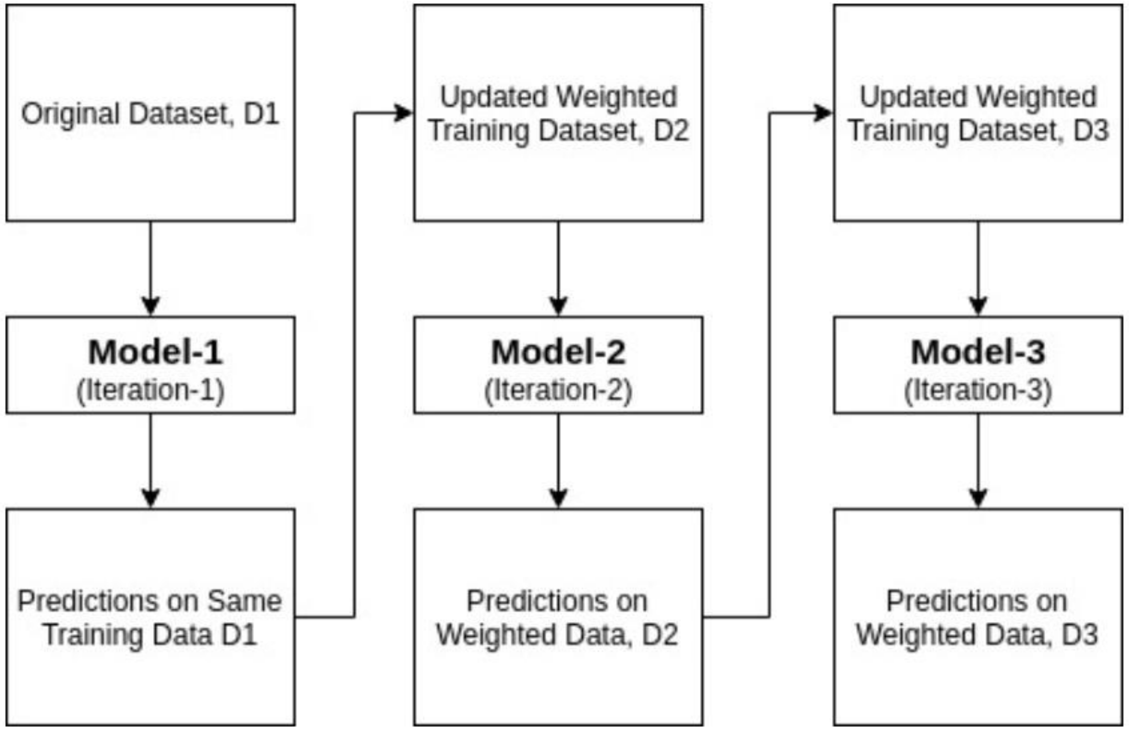
Adaboost

- Ada-boost (Adaptive Boosting)
  - 旨在合併多個分類器以期待創造出擁有更高準確度的分類器
  - 可以結合各種機器學習模型

- 使用Adaboost 的前提
  - 分類器可以在各種不同權重的資料上被訓練
  - 在每次迭代中盡量降低損失

Adaboost

1. 隨機選擇訓練資料集
2. 丟入預測正確的資料作為訓練用
3. 增加錯誤分類資料的權重，確保他們下次會被訓練到
4. 分類越準確的分類器也會得到更高的權重
5. 反覆訓練直到錯誤最小或迭代結束
6. 讓所有模型進行「投票」



AdaBoost

```
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import AdaBoostClassifier
Ada_clf = AdaBoostClassifier(n_estimators=50)
scores = cross_val_score(Ada_clf, data_clean.iloc[:,:-1].values,
data_clean.iloc[:,-1].values)
scores.mean()
```

Gradient Boost 演算法

- Gradient Boost
  - 每一次的計算是為了減少上一次的殘差(residual)，而為了消除殘差，我們可以在殘差減少的梯度(Gradient)方向上建立一個新的模型
  - 每個新的模型皆能使得之前模型的殘差往梯度方向減少

XGBoost

- XGBoost 全名為 Extreme Gradient Boosting
- 梯度提升決策樹 (Gradient Boosted Decision Tree，GBT)
- 應用於解決監督式學習的問題
- XGBoost 優點
  - 記憶體優化：大部分的記憶體分配在第一次加載時就完成，之後便不再進行動態記憶體分配的問題
  - 快取優化：大部份的訓練模式盡可能善用快取機制
  - 改善模型：模型演算法更加強健和更高準確性

評估結果

- Precision：代表所有陽性樣本中，得以正確檢測出陽性結果的機率，以
  TP/(TP+FP)計算
- Recall： 代表所有想抓出來的樣本中，得以正確檢測出陽性結果的機率，以
  TP/(TP+FN)計算
- F1 ：Precision 與 Recall 的調和平均數，以
  2 * Precision * Recall / (Precision + Recall) 計算

	真	假
有	True Positive	False Positive
無	False Negative	True Negative

評估結果

```
from sklearn.metrics import classification_report
print("RF",classification_report(y_test, rfPredict, target_names=None))
print("SVM",classification_report(y_test, predictions_svm, target_names=None))
print("LR",classification_report(y_test, LR_Predict_bin, target_names=None))
print("MLP",classification_report(y_test, predict_NN, target_names=None))
```

THANK YOU