顧客信用預測

李佩怡 108034543

目錄
CONTENT

背景介紹

背景介紹

- 消費習慣改變
- 塑膠貨幣-信用卡崛起
- 先享受後付款-卡奴的產生

透過消費者過去消費行為，做信用分類

資料集介紹

X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	X20	X21	X22	X23	Y
20000	2	2	1	24	2	2	-1	-1	-2	-2	3913	3102	689	0	0	0	0	689	0	0	0	0	1
120000	2	2	2	26	-1	2	0	0	0	2	2682	1725	2682	3272	3455	3261	0	1000	1000	1000	0	2000	1
90000	2	2	2	34	0	0	0	0	0	0	29239	14027	13559	14331	14948	15549	1518	1500	1000	1000	1000	5000	0
50000	2	2	1	37	0	0	0	0	0	0	46990	48233	49291	28314	28959	29547	2000	2019	1200	1100	1069	1000	0
50000	1	2	1	57	-1	0	-1	0	0	0	8617	5670	35835	20940	19146	19131	2000	36681	10000	9000	689	679	0
50000	1	1	2	37	0	0	0	0	0	0	64400	57069	57608	19394	19619	20024	2500	1815	657	1000	1000	800	0
500000	1	1	2	29	0	0	0	0	0	0	367965	412023	445007	542653	483003	473944	55000	40000	38000	20239	13750	13770	0
100000	2	2	2	23	0	-1	-1	0	0	-1	11876	380	601	221	-159	567	380	601	0	581	1687	1542	0
140000	2	3	1	28	0	0	2	0	0	0	11285	14096	12108	12211	11793	3719	3329	0	432	1000	1000	1000	0
20000	1	3	2	35	-2	-2	-2	-2	-1	-1	0	0	0	0	13007	13912	0	0	0	13007	1122	0	0
200000	2	3	2	34	0	0	2	0	0	-1	11073	9787	5535	2513	1828	3731	2306	12	50	300	3738	66	0
260000	2	1	2	51	-1	-1	-1	-1	-1	2	12261	21670	9966	8517	22287	13668	21818	9966	8583	22301	0	3640	0
630000	2	2	2	41	-1	0	-1	-1	-1	-1	12137	6500	6500	6500	6500	2870	1000	6500	6500	6500	2870	0	0
70000	1	2	2	30	1	2	2	0	0	2	65802	67369	65701	66782	36137	36894	3200	0	3000	3000	1500	0	1
250000	1	1	2	29	0	0	0	0	0	0	70887	67060	63561	59696	56875	55512	3000	3000	3000	3000	3000	3000	0
50000	2	3	3	23	1	2	0	0	0	0	50614	29173	28116	28771	29531	30211	0	1500	1100	1200	1300	1100	0
20000	1	1	2	24	0	0	2	2	2	2	15376	18010	17428	18338	17905	19104	3200	0	1500	0	1650	0	1
320000	1	1	1	49	0	0	0	-1	-1	-1	253286	246536	194663	70074	5856	195599	10358	10000	75940	20000	195599	50000	0

資料來源：UCI machine learning
http://archive.ics.uci.edu/ml/datasets/

30000筆資料，23個特徵值x，預測y

特徵值介紹

01 30000筆資料，23364筆如期還款，6636筆延遲還款

02

X1：信用卡額度

X2：性別

X3：教育程度

X4：婚姻狀況

X5：年齡

X6-X11：過去付款情況 [-1,+9]

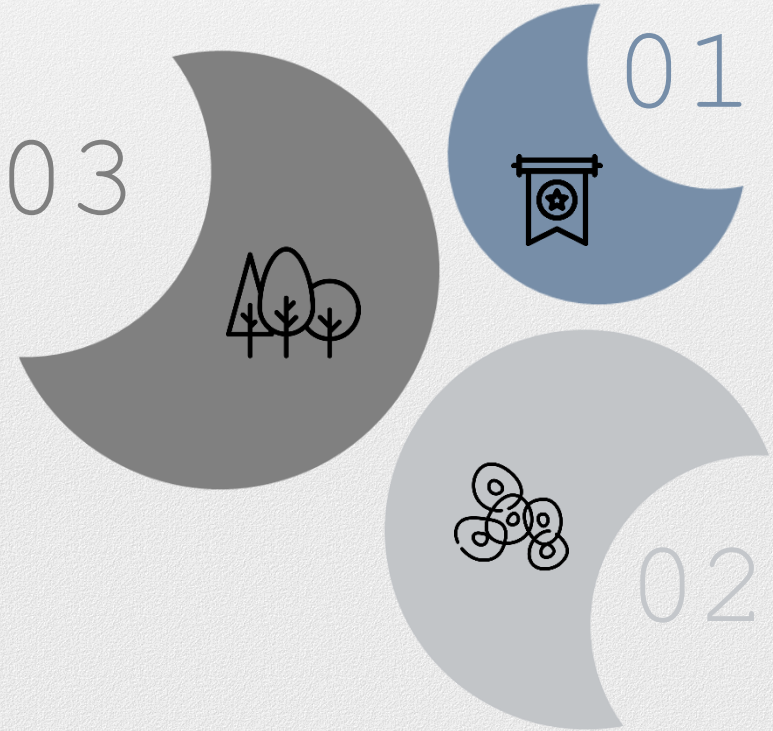
X12-X17：過去帳單金額

X19-23：過去還款金額

X24：預測是否如期償還

資料前處理

資料前處理



01 正規化

02 過取樣（smote）

03 特徵值選取（隨機森林）

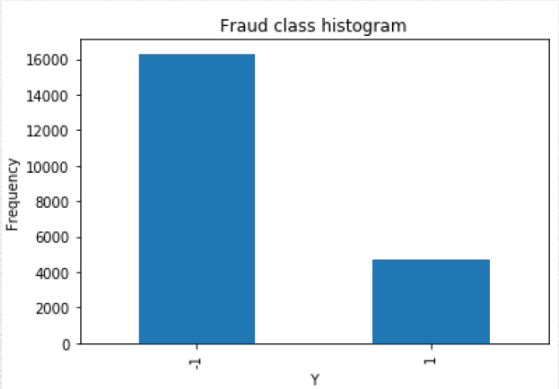
資料前處理-正規化

```
from sklearn.preprocessing import MinMaxScaler
import pandas as pd
data = pd.read_csv('original.csv')  #讀取原始資料檔案
scaler = MinMaxScaler(feature_range=(-1, 1)) #將資料範圍縮到-1到1之間
data_norm= scaler.fit_transform(data)
print(data_norm)

import csv #將正規化後的資料會出成csv檔
data_norm = pd.DataFrame(data_norm)
data_norm.to_csv('data_norm.csv',index = False)
```

- 使用MinMaxScaler套件
- 將資料值分布範圍限制在-1到1之間
- 可避免數值較大的屬性成為關鍵屬性

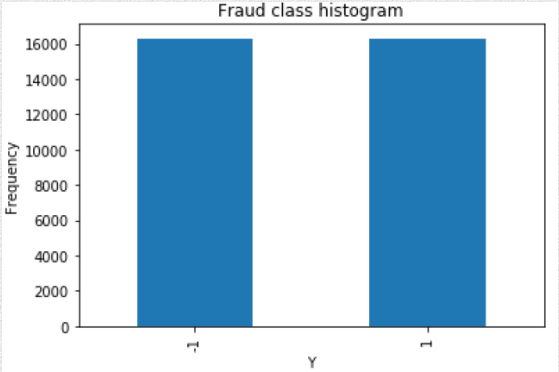
資料前處理-過取樣

```
X = np.array(data.ix[:, data.columns != 'Y'])
y = np.array(data.ix[:, data.columns == 'Y'])
print('Shape of X: {}'.format(X.shape))
print('Shape of y: {}'.format(y.shape))

from imblearn.over_sampling import SMOTE

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

print("Number transactions X_train dataset: ", X_train.shape)
print("Number transactions y_train dataset: ", y_train.shape)
print("Number transactions X_test dataset: ", X_test.shape)
print("Number transactions y_test dataset: ", y_test.shape)
```



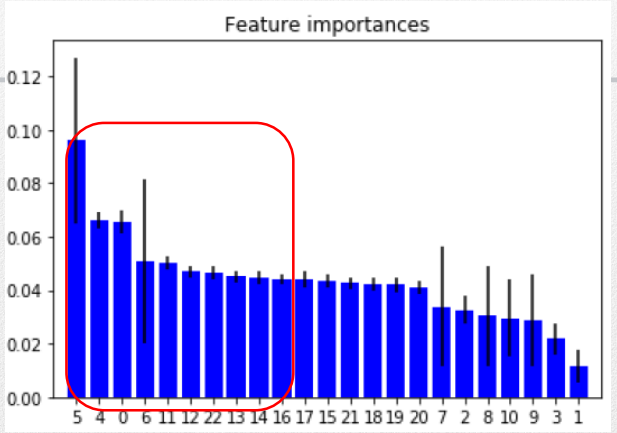
- 使用SMOTE套件
- 將資料值分成30%測試、70%訓練並做過取樣

- 兩類皆為16304筆，資料平衡



資料前處理-特徵值選取

```
tr = RandomForestClassifier() #使用隨機森林分類器
fit = tr.fit(X, y)

importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]

# Print the feature ranking
print("Feature ranking:")

for f in range(X.shape[1]):
    print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indices[f]]))

# Plot the feature importances of the forest
plt.figure() #畫出長條圖
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices],
       color="b", yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), indices)
plt.xlim([-1, X.shape[1]])
plt.show()
#選擇出特徵值輸出存入data
data=df.iloc[:,indices[0:10]]
print('Selected Features:','\n',data.columns,'\n',data)
```




```
Feature ranking:
1. feature 5 (0.096054)
2. feature 4 (0.066378)
3. feature 0 (0.065619)
4. feature 6 (0.050899)
5. feature 11 (0.050399)
6. feature 12 (0.046844)
7. feature 22 (0.046461)
8. feature 13 (0.045059)
9. feature 14 (0.044771)
10. feature 16 (0.044064)
11. feature 17 (0.043688)
12. feature 15 (0.043550)
13. feature 21 (0.042619)
14. feature 18 (0.042169)
15. feature 19 (0.042101)
16. feature 20 (0.040959)
17. feature 7 (0.033914)
18. feature 2 (0.032599)
19. feature 8 (0.030316)
20. feature 10 (0.029489)
21. feature 9 (0.028565)
22. feature 3 (0.021776)
23. feature 1 (0.011505)
```

- 使用RandomForestClassifier套件
- 根據排名選出前10個重要屬性

可避免大量屬性造成維度災難

方法介紹&實際應用

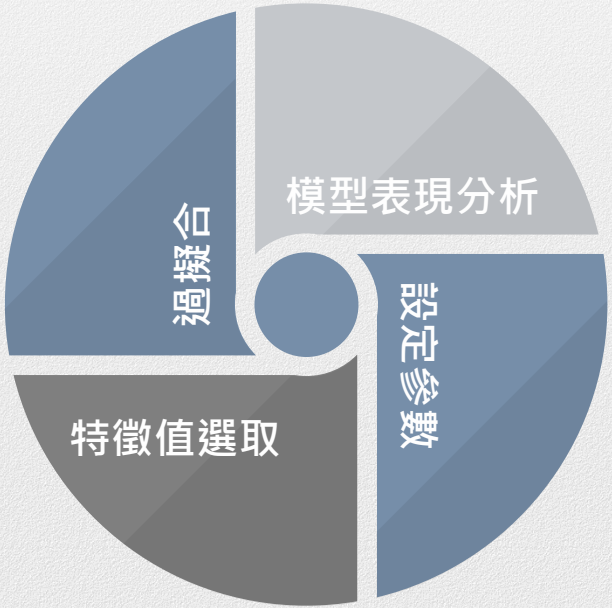
SVM

Smote

30%測試、70%訓練且
做oversampling

Random Forest

引入套件，找出
和預測結果相關
的前10個屬性



Confusion matrix

藉由混淆矩陣評估
準確度、精確度；
召回率等等

SVM kernel

經過嘗試，將
kernel設為rbf

SVM

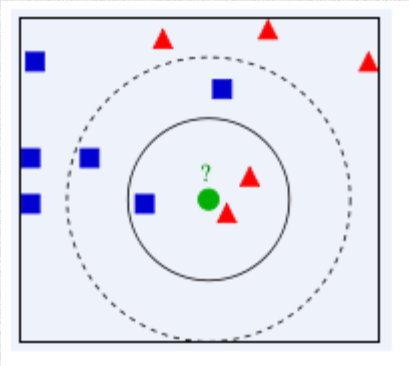
```
import pandas as pd

df1 = pd.read_csv("data_selected.csv")
df1.head()
df1.shape
X = df1.drop('Y', axis=1)
y = df1['Y']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30)

from sklearn.svm import SVC
svclassifier = SVC(kernel='rbf')
svclassifier.fit(X_train, y_train)

y_pred = svclassifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

SVM	正規化	無正規化
所有特徵	0.81	0.78
特徵選擇	0.81	0.79

- 有正規化優於無正規化
- 特徵值選擇對於結果影響不大

KNN(K近鄰分類)

- 根據不同特徵值間的距離進行分類
- 需用於有標籤資料
- 舉例



- K：找最近的K個鄰居
- 若K=3，綠色圓形分到紅色三角形類
- 若K=5，綠色圓形分到藍色正方形類

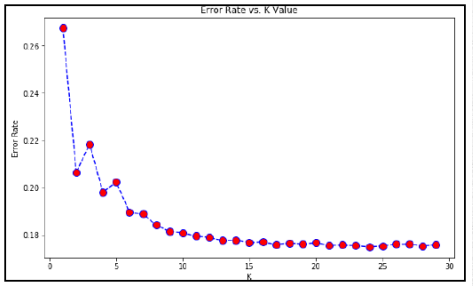
KNN-模型建立

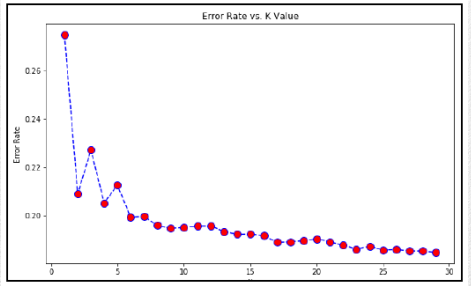
```
scaler = StandardScaler() #載入標準化比例尺套件
scaler.fit(df.drop('Y',axis=1))
scaled_features = scaler.transform(df.drop('Y',axis=1))
```

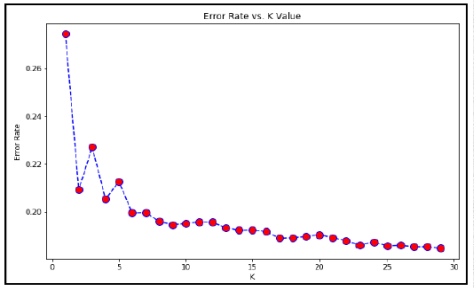
```
for i in range(1,30): #原本只建立K=1的模型 現在建立K為1到30的迴圈

    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))


#將k=1~30的錯誤率製圖畫出，k=26之後，錯誤率就在10%左右震盪，
plt.figure(figsize=(10,6))
plt.plot(range(1,30),error_rate,color='blue', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

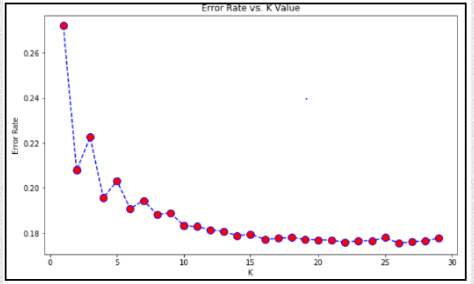
```
knn = KNeighborsClassifier(n_neighbors=26) #選擇錯誤率最低的k=26來看confusionmatrix中各個值的表現

knn.fit(X_train,y_train)
pred = knn.predict(X_test)

print('WITH K=26')
print('\n')
print(confusion_matrix(y_test,pred))
print('\n')
print(classification_report(y_test,pred))
```

KNN-結果







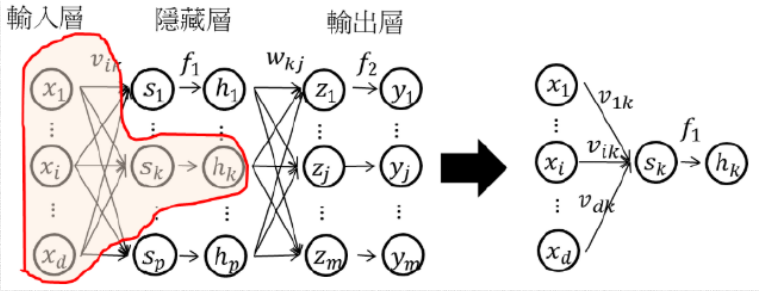


KNN	正規化	無正規化
所有特徵	0.813	0.813
特徵選擇	0.823	0.823

- K=20後，錯誤率不太改變，用它做混淆矩陣
- 由於引入標準話比例尺套件StandardScalar，有無正規化對結果無影響

MLP

- 前向傳遞的神經網路



- 分為輸入層、隱藏層、輸出層
- 每一個神經元都連接到下一層
- 除了輸入節點，其它都有非線性激活函數

MLP-模型建立

- 隱藏層三層，各有10個神經元
- 最大迭代次數300次、容忍值0.000001
- 優化方式為sgd(隨機梯度下降法)

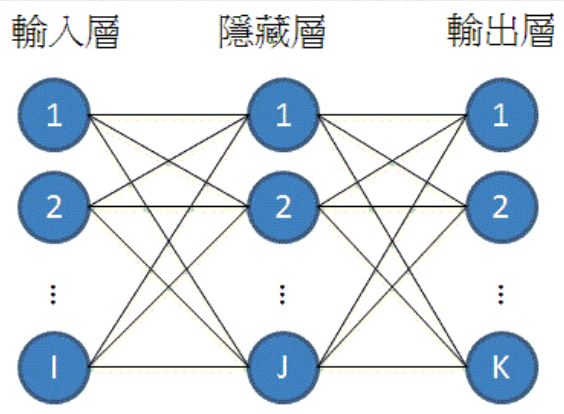
連續兩次迭代超越tol值、或是到達300次就結束訓練

MLP	正規化	無正規化
所有特徵	0.817	0.778
特徵選擇	0.817	0.777

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

y = data['Y']
x = data.drop(['Y'], axis=1)

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size= 0.3, random_state=0)

clf = MLPClassifier(hidden_layer_sizes=(10,10,10), max_iter=300, alpha=0.0001,
                    solver='sgd', verbose=2,  random_state=1,tol=0.000000001)

clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
```

BPN

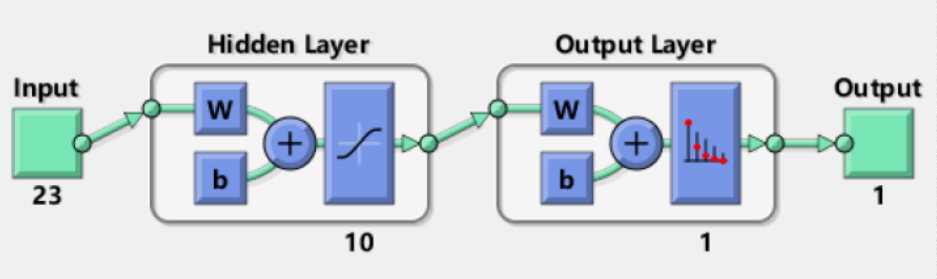
- MLP+EBP(倒傳遞演算法)=BPN(倒傳遞類神經網路)
- 分為向前、向後傳遞



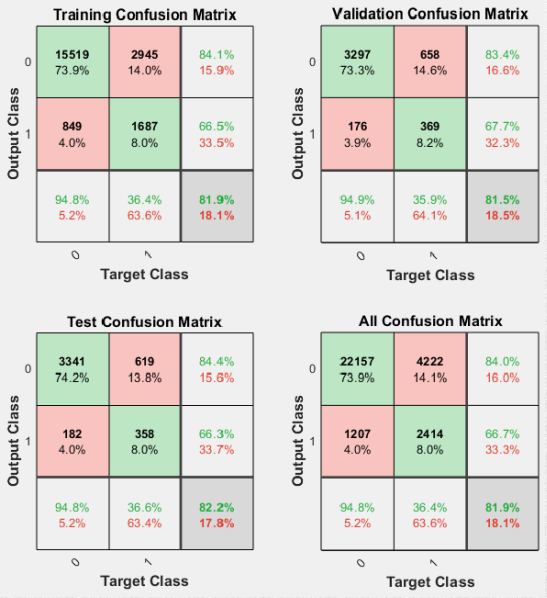
- 向前(左至右)：
  訓練資料丟進模型，計算結果與目標差
- 向後(右至左)：
  根據誤差值調整權重
- 多次訓練後，極小化誤差

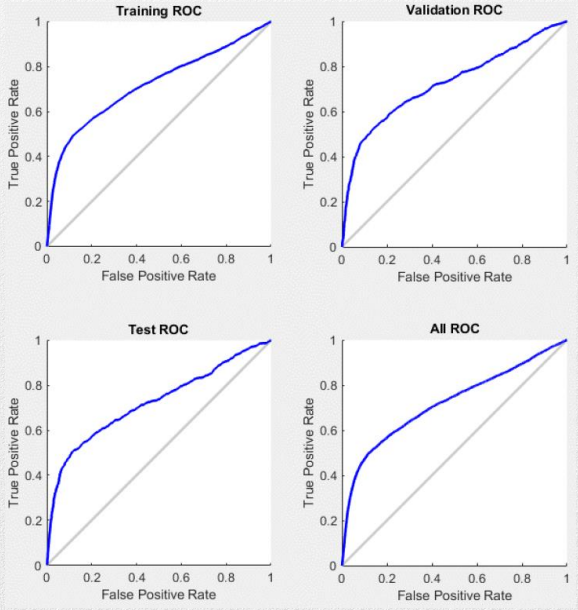
BPN-模型建立

- 使用Matlab
- 將資料分成70%訓練集、30%測試集、再分出15%驗證集
- 神經網路模型



BPN-結果





ROC曲線：

- 預測完全無價值時，呈現45度斜直線(機會線)。
- 若曲線越偏離機會線、曲線下面激越大，表示試驗真實性越佳、準確度越高

BPN	正規化	無正規化
所有特徵	82.2	81.3
特徵選擇	81.9	81.5

結果比較

	SVM	MLP	KNN	BPN
所有特徵	0.81	0.817	0.813	0.822
特徵選取	0.81	0.817	0.823	0.819

- 有正規化優於無正規化
- 所有特徵時，BPN最佳
- 特徵選擇時，KNN最佳

結論

遇到的困難

使用兩個環境可能使得結果無法放在一起比較

調整參數沒有特定方向，只能一直嘗試、效率低

準確度卡住，不知道怎麼更大幅度提升

程式能力不足、且對深度學習了解不夠

THE END