Data Mining final project report

- **組別**：第九組
- **題目**：Default Payments of Credit Card Clients in Taiwan from 2005
- **動機**：以客戶資料及近期刷卡還款紀錄預測未來違規機率
- **資料集敘述**：https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset

欄位說明：

欄位名稱	欄位英文說明	欄位中文說明
ID	ID of each client	客戶 ID
LIMIT_BAL	Amount of given credit in NT dollars (includes individual and family/supplementary credit	客戶的信用額度(包含個人和家庭信用)
SEX	Gender (1=male, 2=female)	客戶性別
EDUCATION	(1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)	教育程度
MARRIAGE	Marital status (1=married, 2=single, 3=others)	婚姻狀況
AGE	Age in years	年齡
PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6	Repayment status within 6 months before October in 2005 (-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, ... 8=payment delay for eight months, 9=payment delay for nine months and above)	前 6 個月的客戶還款狀態
BILL_AMT1, BILL_AMT2, BILL_AMT3, BILL_AMT4, BILL_AMT5, BILL_AMT6	Amount of bill statement within 6 months before October in 2005 (NT dollar)	前 6 個月客戶帳單總額
PAY_AMT1, PAY_AMT2, PAY_AMT3, PAY_AMT4, PAY_AMT5, PAY_AMT6	Amount of previous payment within 6 months before October in 2005 (NT dollar)	前 6 個月客戶還款金額
default.payment.next.month	Default payment (1=yes, 0=no)	下個月是否拖欠

- **分析工具：**

Decision treeRandom

Random Forest

KNeighborsClassifier (10NN)

DecisionTree 及 RandomForest 部分為使用 Scala 來實作，Knn 單純適用 Sklearn.

- **實作與評估方法：**

製作 3 個 model，分別評估其正確率

- **流程：**

```
Import pandas as pd

#----------讀取檔案----------
creditcard = pd.read_csv("UCI_Credit_Card.csv", encoding = 'ISO-8859-1')
df = creditcard.copy()

#----------資料前置處理----------
df = df.drop('ID',axis = 1)

df.EDUCATION = df.EDUCATION.map({1:1,2:2,3:3,4:4,5:0,6:0,0:0})#0視為unknown資料
df.MARRIAGE = df.MARRIAGE.map({0:0,1:1,2:2,3:3})#0視為unknown資料

df = pd.get_dummies(df, columns=['SEX'])
df = pd.get_dummies(df, columns=['EDUCATION'])
df = pd.get_dummies(df, columns=['MARRIAGE'])
df = pd.get_dummies(df, columns=['PAY_0'])
df = pd.get_dummies(df, columns=['PAY_2'])
df = pd.get_dummies(df, columns=['PAY_3'])
df = pd.get_dummies(df, columns=['PAY_4'])
df = pd.get_dummies(df, columns=['PAY_5'])
df = pd.get_dummies(df, columns=['PAY_6'])
```

→將原始資料表讀入並進行資料前處理

→前處理：

1) 刪除 ID 欄位。
2) Education 及 Marriage 出現一些值為非定義，都將其 Mapping 到 0(unknown)。
3) 將 Sex, Education, Marriage, 及前六個月還款狀態處理成 dummy。

```
#----------decision tree----------
x = df.drop('default.payment.next.month', axis=1)
y = df['default.payment.next.month']
```

→定義 x 為 features，y 為 target 部分。

```
from sklearn.datasets import dump_svmlight_file
dump_svmlight_file(x, y, 'svm-output.libsvm')  # where is your y?
from sklearn.datasets import load_svmlight_file


from pyspark.mllib.tree import DecisionTree, DecisionTreeModel
from pyspark.mllib.util import MLUtils
from pyspark.mllib.evaluation import MulticlassMetrics
```

→存為 libsvm 格式，Import 一些套件。

→第一個方法使用 Decision Tree。

```
data  = MLUtils.loadLibSVMFile(sc,"svm-output.libsvm")
(trainingData, testData) = data.randomSplit([0.75, 0.25])
model = DecisionTree.trainClassifier(trainingData, numClasses=2, categoricalFeaturesInfo={},impurity='gini', maxDepth=5, maxBins=32)
```

→載入 libsvm 檔。

→切割 75%為 TrainingData，25% TestingData。

→使用 DecisionTree model

```
predictions = model.predict(testData.map(lambda x: x.features))
labelsAndPredictions = testData.map(lambda lp: lp.label).zip(predictions)

#zz=labelsAndPredictions.take(30)
#scoreAndLabels = sc.parallelize(zz)

#算出來怪怪勿 修好了！----------------------
metrics = MulticlassMetrics(labelsAndPredictions)
precision = metrics.precision(label=1)
recall = metrics.recall(label=1)
Accuracy = metrics.accuracy
print("decision tree Accuracy = %s" % Accuracy)
#算出來怪怪勿----------------------
```

```
decision tree Accuracy = 0.821725154032
```

→計算正確率。

→正確率約為 0.82

```
#--------------------RandomForest----------------------------------------
from pyspark.mllib.tree import RandomForest, RandomForestModel
from pyspark.mllib.util import MLUtils

model = RandomForest.trainClassifier(trainingData, numClasses=2, categoricalFeaturesInfo={},
                                     numTrees=3, featureSubsetStrategy="auto",
                                     impurity='gini', maxDepth=4, maxBins=32)

# Evaluate model on test instances and compute test error
predictions = model.predict(testData.map(lambda x: x.features))
labelsAndPredictions = testData.map(lambda lp: lp.label).zip(predictions)


#print('Learned classification forest model:')
#print(model.toDebugString())

# Save and load model
#model.save(sc, "target/tmp/myRandomForestClassificationModel")
#sameModel = RandomForestModel.load(sc, "target/tmp/myRandomForestClassificationModel")

metrics = MulticlassMetrics(labelsAndPredictions)
precision = metrics.precision(label=1)
recall = metrics.recall(label=1)
Accuracy = metrics.accuracy
print("RandomForest Accuracy = %s" % Accuracy)
```

```
RandomForest Accuracy = 0.80806321993
```

→建立 Random Forest Model，並限制其深度為 4

→正確率約為 0.81

```
#--------------------10NN----------------------------------------
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(x, y,train_size=0.75,test_size=0.25)
neigh = KNeighborsClassifier(n_neighbors=10)
neigh=neigh.fit(X_train, y_train)
neigh_predict = neigh.predict(X_test)
print('10NN accuracy =',accuracy_score(y_test, neigh_predict))
```

```
('10NN accuracy =', 0.7698666666666667)
```

→建立 KNN Model，K=10，資料集中 0.75 作 training data、0.25 作 testing data

→正確率約為 0.76

- 分析結果與結論：

```
In [3]: collections.Counter(y)
Out[3]: Counter({0: 23364, 1: 6636})
```

```
In [5]: 23364/30000
Out[5]: 0.7788
```

原資料集中，target 的正負資料比例相差滿大的，30000 比資料中，23364 為客戶為不會拖欠。因此，模型建構出來要高於 77.8%才有意義，而使用三種方法的結果是使用 KNN 無法有效預測客戶是否會拖欠。

利用 Decision Tree 以及 Random Forest 建立出來的模型之正確率相對較高，可改善判斷客戶是否拖欠，進而採取措施，如：簡訊提醒、專員特別注意等等；亦能作為該客戶是否能使用更多服務之依據，如再辦理其他種類信用卡、借款等服務。