巨量資料分析

Paper Study 3

日期：2021/12/9

組別：

姓名及分工：

**繳交方式：**12/9 下午14:00 之前於E3系統繳交。

**注意事項：**程式或軟體的截圖放在附錄，數值資料經過整理之後可以放在作業的正文。

**研讀論文：**

請各組研讀論文，並且回答下列問題，回答整理在MS Word報告。

Sakar, C.O., Polat, S.O., Katircioglu, M., Kastro, Y., Real-time prediction of online shoppers’ purchasing intention using multilayer perceptron and LSTM recurrent neural networks. Neural Computation & Application, 31, 6893–6908 (2019). https://doi.org/10.1007/s00521-018-3523-0

本篇論文的dataset可以從Data Sets - UCI Machine Learning Repository下載，其下載連結如下列所示。

Online Shoppers Purchasing Intention Dataset Data Set

<https://archive.ics.uci.edu/ml/datasets/Online+Shoppers+Purchasing+Intention+Dataset>

**問題：**

1. 說明本篇論文的研究目的。
2. 說明本篇論文的dataset。
3. 說明本篇論文dataset的屬性與分類的類別變數。
4. 本篇論文應用oversampling擬解決甚麼問題？
5. 說明本篇論文之oversampling的方式。
6. 說明本篇論文中Tables 3-5整理的結果之採用的方法(方法名稱)。
7. 說明本篇論文中Tables 6-8整理的結果之採用的方法(方法名稱)。
8. 使用CART方法針對Data Sets - UCI Machine Learning Repository下載的dataset建立分類模式。將CART的參數設定整理表格說明，並且整理如 Tables 3-8的績效指標。
9. 使用Gradient Boosted Tree方法針對Data Sets - UCI Machine Learning Repository下載的dataset建立分類模式。將Gradient Boosted Tree的參數設定整理表格說明，並且整理如 Tables 3-8的績效指標。
10. 使用Rule-based方法(例如，Rule Induction)針對Data Sets - UCI Machine Learning Repository下載的dataset建立分類模式。將Rule-based方法的參數設定整理表格說明，並且整理如 Tables 3-8的績效指標。

由下一頁依據問題順序回答，附錄整理在問題11的回答之後。

5.

先將原始資料的30%當作testing set，再把剩下的70%作為training set，並針對其少數的label進行重複取樣來增加

6. Cross validation on Testing set  
Table 3: C4.5、Random Forest

Table4: Multilayer perceptron (MLP)

Table5: Support Vector Machine (SVM) with Linear and Radical basis function kernel

Evaluate by using Accuracy、TPR、TNR、F1 Score

7. Cross validation on Testing set with oversampling

Table 3:C4.5、Random Forest

Table4: Multilayer perceptron (MLP)

Table5: Support Vector Machine (SVM) with Linear and Radical basis function kernel

Evaluate by using Accuracy、TPR、TNR、F1 Score

9.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classifier | Accuracy(%) | True-positive rate (TPR) | True-negative rate (TNR) | F1 Score |
| Gradient Boost Classifier | 90.67 | 0.62 | 0.95 | 0.67 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classifier (oversample) | Accuracy(%) | True-positive rate (TPR) | True-negative rate (TNR) | F1 Score |
| Gradient Boost Classifier | 85.86 | 0.85 | 0.86 | 0.64 |

10.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classifier | Accuracy(%) | True-positive rate (TPR) | True-negative rate (TNR) | F1 Score |
| C4.5 | 84.75 | 0.52 | 0.90 | 0.51 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classifier (oversample) | Accuracy(%) | True-positive rate (TPR) | True-negative rate (TNR) | F1 Score |
| C4.5 | 83.40 | 0.62 | 0.87 | 0.53 |

附錄

import pandas as pd

import numpy as np

from sklearn.ensemble import GradientBoostingClassifier

from chefboost import Chefboost as chef

from sklearn.model\_selection import train\_test\_split

from sklearn import metrics

from sklearn.metrics import confusion\_matrix

import seaborn as sns

import matplotlib.pyplot as plt

from tqdm import tqdm

#%%

data=pd.read\_csv("online\_shoppers\_intention.csv",encoding="utf-8-sig")

dummy\_ls=list(data.columns[-8:-2])

for d in dummy\_ls:

tmp\_df=pd.get\_dummies(data[d],prefix=d+"\_",drop\_first=True)

data=pd.concat([data,tmp\_df],axis=1)

del data[d]

data["weekend"]=np.where(data["Weekend"]==True,1,0)

del data["Weekend"]

data["revenue"]=np.where(data["Revenue"]==True,1,0)

del data["Revenue"]

# data.to\_csv("paper\_study3.csv",encoding="utf-8-sig")

x=data.drop(columns=["revenue"])

y=data["revenue"]

data["revenue"].value\_counts()

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.3, random\_state=1207)

#%%

clf=GradientBoostingClassifier(random\_state=1130,max\_depth=2 )

clf.fit(x\_train, y\_train)

train\_pred=clf.predict(x\_train)

print("Accuracy:",metrics.accuracy\_score(y\_train, train\_pred))

test\_pred=clf.predict(x\_test)

print("Accuracy:",metrics.accuracy\_score(y\_test, test\_pred))

cm = confusion\_matrix(y\_test, test\_pred)

sns.heatmap(cm, annot=True, fmt='d',cmap='YlGnBu')

plt.ylabel("True Value", color = 'r')

plt.xlabel("Predicted Value", color = 'r')

tn=cm[0][0]

tp=cm[1][1]

fn=cm[1][0]

fp=cm[0][1]

print("TPR:", tp/(tp+fn))

print("TNR:", tn/(tn+fp))

print("F1-score:",metrics.f1\_score(y\_test,test\_pred))

#%%

train=pd.concat([x\_train,y\_train],axis=1)

config={"algorithm":"C4.5","enableParallelism":True, "num\_cores":8}

train["Decision"]=np.where(train["revenue"]==1,"Yes","No")

del train["revenue"]

model=chef.fit(train,config)

chef\_test\_pred=[]

for i in tqdm(range(len(x\_test))):

pred=chef.predict(model,x\_test.iloc[i])

if pred=="No":

chef\_test\_pred.append(0)

else:

chef\_test\_pred.append(1)

#%%

chef\_cm=confusion\_matrix(y\_test,chef\_test\_pred)

sns.heatmap(chef\_cm, annot=True, fmt='d',cmap='YlGnBu')

plt.ylabel("True Value", color = 'r')

plt.xlabel("Predicted Value", color = 'r')

tn=chef\_cm[0][0]

tp=chef\_cm[1][1]

fn=chef\_cm[1][0]

fp=chef\_cm[0][1]

print("TPR:", tp/(tp+fn))

print("TNR:", tn/(tn+fp))

print("F1-score:",metrics.f1\_score(y\_test,chef\_test\_pred))

print("Accuracy:",metrics.accuracy\_score(y\_test, chef\_test\_pred))