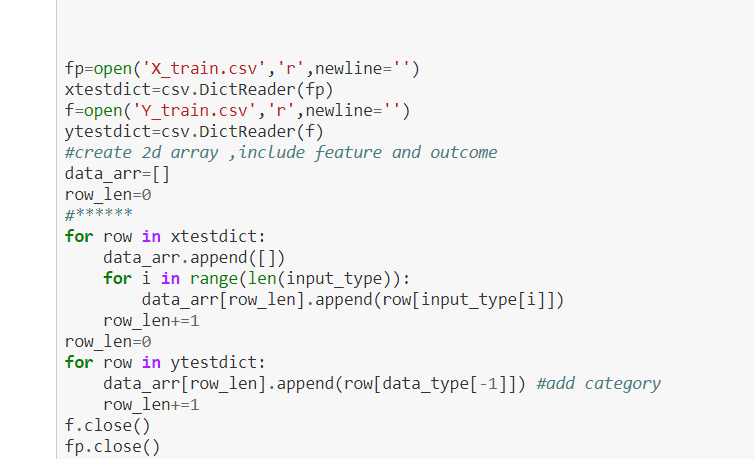
Hw2 report

1. data input:

import csv package用csv.Reader讀csv檔，再把讀出來的dictionary轉成二維list儲存，把X\_train.csv跟Y\_train.csv存到同個2維list data\_arr裡。



1. data preprocessing

先用random.shuffle() shuffle data\_arr讓裡面的資料亂數排列，再把有missing value的資料drop掉，在用trans\_string\_to\_codestring()把category的features轉成用數字代表各種類別，再用trans\_continue\_to\_bounding()把continue features轉成用15個數字把連續資料bounding並代表在哪個區間的資料。



3 .model construction:

Decision tree:

利用crete\_tree() create decision tree，先判斷是否到了leaf，如果不是，用best\_attr,threshold = choose\_attr(data\_arr)選出要以此分類的attribute跟要用哪個數字當分類的threshold，因為所有attribute都被轉成用數字表示，所以可以用threshold來區分，如果小於等於threshold就是左節點，大於threshold就是柚節點，詳細程式碼如下

feature\_arr=['age','workclass','fnlwgt','education','education-num',

'marital-status','occupation','relationship','race','sex',

'capital-gain','capital-loss','hours-per-week','native-country']

input\_type=['Id','age','workclass','fnlwgt','education','education-num',

'marital-status','occupation','relationship','race','sex',

'capital-gain','capital-loss','hours-per-week','native-country']

feature\_continus\_discrete\_arr=['c','d','c','d','c','d','d','d','d','d','c','c','c','d']

data\_type=['Id','age','workclass','fnlwgt','education','education-num',

'marital-status','occupation','relationship','race','sex',

'capital-gain','capital-loss','hours-per-week','native-country',

'Category']

continue\_feature\_threshold={'age':37,'fnlwgt':178233,'education-num':10,'capital-gain':0,'capital-loss':0,'hours-per-week':40}

*#判斷標準 中位數*

answer\_arr=['Id','Category']

**class** **Node**(object):

**def** \_\_init\_\_(self,attribute,threshold):

self.attr = attribute

self.thres = threshold

self.left = **None**

self.right = **None**

self.leaf = **False**

self.predict = **None**

**def** trans\_continue\_to\_bounding(data\_arr,n):

**for** i **in** range(1,len(feature\_arr)+1):

**if** feature\_continus\_discrete\_arr[i-1]=='d':

**continue**

values=[int(row[i]) **for** row **in** data\_arr]

values.sort()

size,rem=div\_list\_num(values,n)

start=0

end=size

**for** row **in** data\_arr:

**for** j **in** range(n):

start=j\*size

end=start+size-1

**if** j==(n-1):

end+=rem

**if** int(row[i]) >= values[start] **and** int(row[i]) <=values[end]:

row[i]=str(j)

**def** div\_list\_num(values,n):

size=0

size+=math.floor(len(values)/n)

rem=len(values)-size\*n

**while** rem > n:

size+=math.floor(rem/n)

rem=len(values)-size\*n

**return** size,rem

**def** trans\_string\_to\_codestring(data\_arr):

**for** i **in** range(1,len(feature\_arr)+1):

**if** feature\_continus\_discrete\_arr[i-1]=='c':

**continue**

codedict={}

num=0

**for** row **in** range(len(data\_arr)):

**if** data\_arr[row][i] **not** **in** codedict.keys():

codedict[data\_arr[row][i]]=num

num+=1

data\_arr[row][i]=str(codedict[data\_arr[row][i]])

**def** compute\_entropy(data\_arr):

arr\_len=len(data\_arr)

labelcount={}

**for** row **in** data\_arr:

tmp\_category=row[-1]

**if** tmp\_category **not** **in** labelcount.keys():

labelcount[tmp\_category]=0

labelcount[tmp\_category]+=1

entro=0

*#print(labelcount)*

**for** key **in** labelcount:

pro=float(labelcount[key])/arr\_len

entro -=pro\*math.log(pro,2)

**return** entro

**def** find\_most\_category(classlist):

classcount={}

**for** vote **in** classlist:

**if** vote **not** **in** classcount.keys():

classcount[vote]=0

classcount[vote]+=1

sortedlist= sorted(classcount.items(),key=operator.itemgetter(1),reverse=**True**)

*#print(type(sortedlist))*

**return** sortedlist[0][0]

**def** choose\_thres(data\_arr,attribute):

**if**(feature\_continus\_discrete\_arr[attribute-1]=='d'):

values=[float(row[attribute]) **for** row **in** data\_arr]

values=set(values)

values=list(values)

values.sort()

max\_ig=float("-inf")

thres\_val=0

**for** i **in** range(0,len(values)-1):

thres=(values[i]+values[i+1])/2

ig=info\_gain(data\_arr,attribute,thres)

**if** ig>max\_ig:

max\_ig=ig

thres\_val=thres

**return** thres\_val

**else**:

values=[float(row[attribute]) **for** row **in** data\_arr]

**return** np.median(values)

**def** info\_gain(data\_arr,attr,threshold):

sub1=[row **for** row **in** data\_arr **if** float(row[attr])<=threshold]

*#print("sub1len:",len(sub1))*

sub2=[row **for** row **in** data\_arr **if** float(row[attr])>threshold]

*#print("sub2len:",len(sub2))*

ig=compute\_entropy(data\_arr)-remainder(data\_arr,[sub1 ,sub2])

**return** ig

**def** remainder(data\_arr,data\_subsets):

num=len(data\_arr)

rem=float(0)

**for** sub **in** data\_subsets:

rem += float(len(sub)/num)\*compute\_entropy(sub)

**return** rem

**def** choose\_attr(data\_arr):

max\_info\_gain=float('-inf')

best\_attr=**None**

threshold=0

**for** attr **in** range(1,len(feature\_arr)+1 ):

thres=choose\_thres(data\_arr,attr)

ig = info\_gain(data\_arr,attr,thres)

**if** ig > max\_info\_gain:

max\_info\_gain=ig

best\_attr=attr

threshold=thres

**return** best\_attr,threshold

**def** create\_tree(data\_arr):

*#print("now arrlen:",len(data\_arr))*

classlist=[row[-1] **for** row **in** data\_arr]

**if** classlist.count(classlist[0])==len(classlist):

leaf = Node(**None**,**None**)

leaf.leaf=**True**

leaf.predict=classlist[0]

**return** leaf

best\_attr,threshold = choose\_attr(data\_arr)

attrlist=[row[best\_attr] **for** row **in** data\_arr ]

**if** attrlist.count(attrlist[0])==len(attrlist):

leaf = Node(**None**,**None**)

leaf.leaf=**True**

f=classlist.count('0')

t=classlist.count('1')

**if** t>f:

leaf.predict='1'

**else**:

leaf.predict='0'

**return** leaf

*#print("choose success")*

tree=Node(best\_attr,threshold)

sub1=[row **for** row **in** data\_arr **if** float(row[best\_attr]) <= threshold]

sub2=[row **for** row **in** data\_arr **if** float(row[best\_attr]) > threshold]

*#print("sub1len:",len(sub1))*

*#print("sub2len:",len(sub2))*

*#print("attribute:",feature\_arr[best\_attr-1],'threshold:',threshold)*

tree.left = create\_tree(sub1)

tree.right = create\_tree(sub2)

random forest:

用crete\_forest()建，qautity代表建幾顆tree，data\_size代表要建一顆tree所需資料量，回傳由一堆tree的root組成的list。

**def** create\_forest(data\_arr,quantity,sub\_data\_size):

forest=[]

**for** i **in** range(quantity):

sub\_arr=random.sample(data\_arr,sub\_data\_size)

forest.append(create\_tree(sub\_arr))

**return** forest

4validation

實作在main function裡，基本上就是調整傳進create\_tree或create\_forest裡的參數，跟result的計算要稍微改一下，更加詳細情形在hw2-1.ipynb裡。

5.result

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value>=thres 0.5 go to right node

in the leaf the prediction is: 0

attribe: native-country the value>=thres 1.5 go to right node

attribe: native-country the value<=thres 18.5 go to left node

attribe: native-country the value<=thres 4.5 go to left node

attribe: native-country the value<=thres 2.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value>=thres 1.5 go to right node

attribe: native-country the value<=thres 18.5 go to left node

attribe: native-country the value<=thres 4.5 go to left node

attribe: native-country the value>=thres 2.5 go to right node

attribe: native-country the value<=thres 3.5 go to left node

in the leaf the prediction is: 0

decision tree with holdout validation result:

confusion matrix

true positive: 4753 false positive: 1570

false negative: 11 true negative: 5

Accuracy: 0.750591575958353

Precision: 0.751700142337498

Recall: 0.9976910159529807

random forest has2000 tree with 50 datas with holdout validation result:

confusion matrix

true positive: 4764 false positive: 1575

false negative: 0 true negative: 0

Accuracy: 0.751538097491718

Precision: 0.751538097491718

Recall: 1.0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value>=thres 0.5 go to right node

in the leaf the prediction is: 0

attribe: native-country the value>=thres 1.5 go to right node

attribe: native-country the value<=thres 18.5 go to left node

attribe: native-country the value<=thres 4.5 go to left node

attribe: native-country the value<=thres 2.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value>=thres 1.5 go to right node

attribe: native-country the value<=thres 18.5 go to left node

attribe: native-country the value<=thres 4.5 go to left node

attribe: native-country the value>=thres 2.5 go to right node

attribe: native-country the value<=thres 3.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 2.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 2.5 go to left node

attribe: native-country the value>=thres 0.5 go to right node

attribe: native-country the value<=thres 1.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 2.5 go to left node

attribe: native-country the value>=thres 0.5 go to right node

attribe: native-country the value>=thres 1.5 go to right node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 2.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 2.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 2.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 2.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 2.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 2.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value>=thres 2.5 go to right node

attribe: native-country the value<=thres 4.5 go to left node

attribe: native-country the value<=thres 3.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value>=thres 0.5 go to right node

in the leaf the prediction is: 0

attribe: native-country the value>=thres 1.5 go to right node

attribe: native-country the value<=thres 4.5 go to left node

attribe: native-country the value<=thres 2.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value<=thres 1.5 go to left node

attribe: native-country the value<=thres 0.5 go to left node

in the leaf the prediction is: 0

attribe: native-country the value>=thres 1.5 go to right node

attribe: native-country the value<=thres 4.5 go to left node

attribe: native-country the value>=thres 2.5 go to right node

attribe: native-country the value<=thres 3.5 go to left node

in the leaf the prediction is: 0

decision tree with k-fold result:

confusion matrix

true positive: 4756.0 false positive: 1570.3333333333333

false negative: 8.0 true negative: 4.666666666666667

Accuracy: 0.7510122521954041

Precision: 0.7517782812582329

Recall: 0.998320738874895

ranndom forest has2000 tree with 50 datas with k-fold validation result:

confusion matrix

true positive: 5276.333333333333 false positive: 1768.0

false negative: 0.0 true negative: 0.0

Accuracy: 0.7490181233142479

Precision: 0.7490181233142479

Recall: 1.0

6.comparsion&result:

這次作業我做得不夠好，如果有更多時間我應該會把我的decision tree改成多節點樹，因為要做2元樹，我把所有feature用數字表示，所以可能被分到同個節點的資料其實之間也沒有相關性，只是數字剛好同時小於\大於threshold而已，而且從我的prediction result可以看出來因為我的feature不會越來越少，導致可能都只用同一個feature來判斷，而且改成多節點樹，深度最多等於feature數量而已，總體來說，我對這次作業並不太滿意，希望下次可以更好，不過網路上的資料真的很難找，超大部分都是用sklearn的，讓我花了好長時間找資料。

7.kaggle submission

