ML skill4

September 2, 2021

[1]:
```
import pandas as pd
import numpy as np
data=pd.read_csv(r'E:\M&L excel\adult_dataset.csv')
data.head()
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             32561 non-null  int64 
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64 
 3   education       32561 non-null  object
 4   education.num   32561 non-null  int64 
 5   marital.status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital.gain    32561 non-null  int64 
 11  capital.loss    32561 non-null  int64 
 12  hours.per.week  32561 non-null  int64 
 13  native.country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

[2]:
```
print(data.isna().sum())
print(data.nunique())
data.describe()
```

```
age               0
workclass         0
fnlwgt            0
education         0
education.num     0
marital.status    0
```

```
occupation        0
relationship      0
race              0
sex               0
capital.gain      0
capital.loss      0
hours.per.week    0
native.country    0
income            0
dtype: int64
age                  73
workclass             9
fnlwgt            21648
education            16
education.num        16
marital.status        7
occupation           15
relationship          6
race                  5
sex                   2
capital.gain        119
capital.loss         92
hours.per.week       94
native.country       42
income                2
dtype: int64
```

[2]:

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

[3]:
```
(data=='?').sum()#missing values
data=data[data['occupation']!='?']
(data=='?').sum()
data=data[data['native.country']!='?']
(data=='?').sum()
```

[3]:
```
age               0
workclass         0
fnlwgt            0
education         0
education.num     0
marital.status    0
occupation        0
relationship      0
race              0
sex               0
capital.gain      0
capital.loss      0
hours.per.week    0
native.country    0
income            0
dtype: int64
```

[4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30162 entries, 1 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             30162 non-null  int64
 1   workclass       30162 non-null  object
 2   fnlwgt          30162 non-null  int64
 3   education       30162 non-null  object
 4   education.num   30162 non-null  int64
 5   marital.status  30162 non-null  object
 6   occupation      30162 non-null  object
 7   relationship    30162 non-null  object
 8   race            30162 non-null  object
 9   sex             30162 non-null  object
 10  capital.gain    30162 non-null  int64
 11  capital.loss    30162 non-null  int64
 12  hours.per.week  30162 non-null  int64
 13  native.country  30162 non-null  object
 14  income          30162 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

[5]:
```
#convert categorical to numerical
from sklearn import preprocessing
categ=data.select_dtypes(include=['object'])
categ=categ.apply(preprocessing.LabelEncoder().fit_transform)
categ['income']=categ['income'].astype('category')
categ.head()
```

[5]:

	workclass	education	marital.status	occupation	relationship	race	sex	native.country	income
1	2	11	6	3	1	4	0	38	0
3	2	5	0	6	4	4	0	38	0
4	2	15	5	9	3	4	0	38	0
5	2	11	0	7	4	4	0	38	0
6	2	0	5	0	4	4	1	38	0

[6]:
```
#update the data with numerical data
data.drop(categ.columns,axis=1,inplace=True)
data=pd.concat([data,categ],axis=1)
data.head()
```

[6]:

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week	workclass	education	marital.status	occupation	relationship	race	sex	native.country	income
1	82	132870	9	0	4356	18	2	11	6	3	1	4	0	38	0
3	54	140359	4	0	3900	40	2	5	0	6	4	4	0	38	0
4	41	264663	10	0	3900	40	2	15	5	9	3	4	0	38	0
5	34	216864	9	0	3770	45	2	11	0	7	4	4	0	38	0
6	38	150601	6	0	3770	40	2	0	5	0	4	4	1	38	0

[7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30162 entries, 1 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             30162 non-null  int64
 1   fnlwgt          30162 non-null  int64
 2   education.num   30162 non-null  int64
 3   capital.gain    30162 non-null  int64
 4   capital.loss    30162 non-null  int64
 5   hours.per.week  30162 non-null  int64
 6   workclass       30162 non-null  int32
 7   education       30162 non-null  int32
 8   marital.status  30162 non-null  int32
 9   occupation      30162 non-null  int32
 10  relationship    30162 non-null  int32
 11  race            30162 non-null  int32
 12  sex             30162 non-null  int32
 13  native.country  30162 non-null  int32
 14  income          30162 non-null  category
dtypes: category(1), int32(8), int64(6)
memory usage: 2.6 MB
```

[21]:
```
from sklearn.model_selection import train_test_split
X=data.drop('income',axis=1)
y=data['income']
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.
 ↪20,random_state=42)
```

[29]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
dt = DecisionTreeClassifier(criterion="entropy")
dt.fit(X_train,y_train)
#tree.plot_tree(dt)
```

[29]:
```
DecisionTreeClassifier(criterion='entropy')
```

[30]:
```
from sklearn import metrics
y_pred=dt.predict(X_test)
print("Accuracy for testing:",metrics.accuracy_score(y_test, y_pred))
ytrain_pred=dt.predict(X_train)
print("Accuracy for training:",metrics.accuracy_score(y_train, ytrain_pred))
print(metrics.classification_report(y_test,y_pred))
```

```
Accuracy for testing: 0.8054036134593071
Accuracy for training: 0.9999585560943264
```

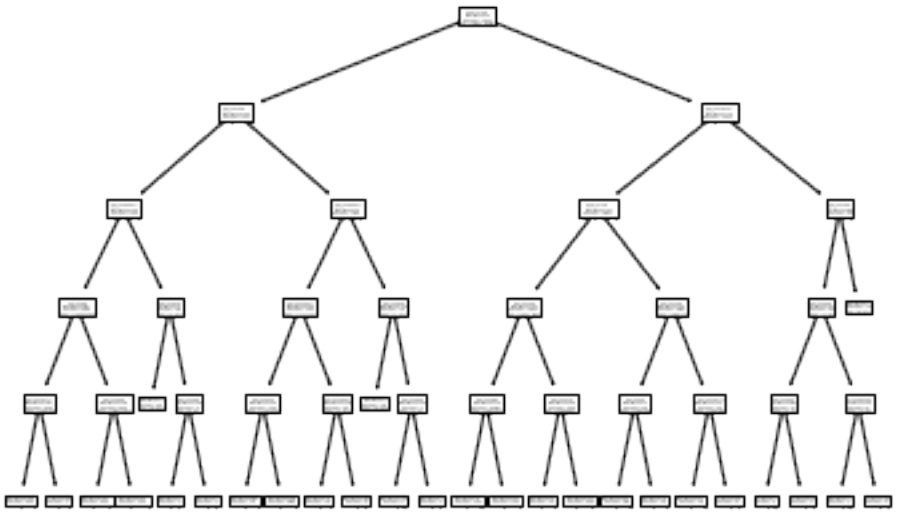
```
              precision    recall  f1-score   support

           0       0.87      0.87      0.87      4533
           1       0.61      0.62      0.61      1500

    accuracy                           0.81      6033
   macro avg       0.74      0.74      0.74      6033
weighted avg       0.81      0.81      0.81      6033
```

[33]:
```
#reduce depth to get accurate predictions
# max_depth which is 5 so that we can plot and read the tree.
dt = DecisionTreeClassifier(criterion="entropy",max_depth=5)
dt.fit(X_train,y_train)
tree.plot_tree(dt)
```

[33]:
```
[Text(177.8625, 199.32, 'X[10] <= 0.5\nentropy = 0.81\nsamples = 24129\nvalue =
[18121, 6008]'),
 Text(87.1875, 163.07999999999998, 'X[2] <= 12.5\nentropy = 0.995\nsamples =
9921\nvalue = [5362, 4559]'),
 Text(45.337500000000006, 126.83999999999999, 'X[3] <= 5095.5\nentropy =
0.926\nsamples = 6956\nvalue = [4584, 2372]'),
 Text(27.900000000000002, 90.6, 'X[2] <= 8.5\nentropy = 0.887\nsamples =
6588\nvalue = [4578, 2010]'),
 Text(13.950000000000001, 54.359999999999985, 'X[4] <= 1791.5\nentropy =
0.467\nsamples = 1036\nvalue = [933, 103]'),
 Text(6.9750000000000005, 18.119999999999976, 'entropy = 0.439\nsamples =
1012\nvalue = [920, 92]'),
 Text(20.925, 18.119999999999976, 'entropy = 0.995\nsamples = 24\nvalue = [13,
11]'),
 Text(41.85, 54.359999999999985, 'X[0] <= 35.5\nentropy = 0.928\nsamples =
5552\nvalue = [3645, 1907]'),
 Text(34.875, 18.119999999999976, 'entropy = 0.757\nsamples = 1822\nvalue =
[1424, 398]'),
 Text(48.825, 18.119999999999976, 'entropy = 0.974\nsamples = 3730\nvalue =
[2221, 1509]'),
 Text(62.775000000000006, 90.6, 'X[0] <= 61.5\nentropy = 0.12\nsamples =
368\nvalue = [6, 362]'),
 Text(55.800000000000004, 54.359999999999985, 'entropy = 0.0\nsamples =
337\nvalue = [0, 337]'),
 Text(69.75, 54.359999999999985, 'X[5] <= 46.5\nentropy = 0.709\nsamples =
31\nvalue = [6, 25]'),
 Text(62.775000000000006, 18.119999999999976, 'entropy = 0.9\nsamples =
19\nvalue = [6, 13]'),
 Text(76.72500000000001, 18.119999999999976, 'entropy = 0.0\nsamples = 12\nvalue
= [0, 12]'),
 Text(129.03750000000002, 126.83999999999999, 'X[3] <= 5095.5\nentropy =
```

```
0.83\nsamples = 2965\nvalue = [778, 2187]'),
 Text(111.60000000000001, 90.6, 'X[4] <= 1782.5\nentropy = 0.895\nsamples =
2492\nvalue = [777, 1715]'),
 Text(97.65, 54.359999999999985, 'X[5] <= 31.0\nentropy = 0.933\nsamples =
2209\nvalue = [771, 1438]'),
 Text(90.67500000000001, 18.119999999999976, 'entropy = 0.864\nsamples =
143\nvalue = [102, 41]'),
 Text(104.62500000000001, 18.119999999999976, 'entropy = 0.908\nsamples =
2066\nvalue = [669, 1397]'),
 Text(125.55000000000001, 54.359999999999985, 'X[4] <= 1989.5\nentropy =
0.148\nsamples = 283\nvalue = [6, 277]'),
 Text(118.575, 18.119999999999976, 'entropy = 0.04\nsamples = 235\nvalue = [1,
234]'),
 Text(132.525, 18.119999999999976, 'entropy = 0.482\nsamples = 48\nvalue = [5,
43]'),
 Text(146.47500000000002, 90.6, 'X[0] <= 62.5\nentropy = 0.022\nsamples =
473\nvalue = [1, 472]'),
 Text(139.5, 54.359999999999985, 'entropy = 0.0\nsamples = 426\nvalue = [0,
426]'),
 Text(153.45000000000002, 54.359999999999985, 'X[0] <= 63.5\nentropy =
0.149\nsamples = 47\nvalue = [1, 46]'),
 Text(146.47500000000002, 18.119999999999976, 'entropy = 0.722\nsamples =
5\nvalue = [1, 4]'),
 Text(160.425, 18.119999999999976, 'entropy = 0.0\nsamples = 42\nvalue = [0,
42]'),
 Text(268.5375, 163.07999999999998, 'X[3] <= 7073.5\nentropy = 0.475\nsamples =
14208\nvalue = [12759, 1449]'),
 Text(223.20000000000002, 126.83999999999999, 'X[10] <= 4.5\nentropy =
0.411\nsamples = 13895\nvalue = [12750, 1145]'),
 Text(195.3, 90.6, 'X[2] <= 12.5\nentropy = 0.294\nsamples = 12833\nvalue =
[12167, 666]'),
 Text(181.35000000000002, 54.359999999999985, 'X[0] <= 28.5\nentropy =
0.179\nsamples = 10188\nvalue = [9914, 274]'),
 Text(174.375, 18.119999999999976, 'entropy = 0.05\nsamples = 4444\nvalue =
[4419, 25]'),
 Text(188.32500000000002, 18.119999999999976, 'entropy = 0.257\nsamples =
5744\nvalue = [5495, 249]'),
 Text(209.25000000000003, 54.359999999999985, 'X[0] <= 27.5\nentropy =
0.605\nsamples = 2645\nvalue = [2253, 392]'),
 Text(202.275, 18.119999999999976, 'entropy = 0.15\nsamples = 792\nvalue = [775,
17]'),
 Text(216.22500000000002, 18.119999999999976, 'entropy = 0.727\nsamples =
1853\nvalue = [1478, 375]'),
 Text(251.10000000000002, 90.6, 'X[2] <= 10.5\nentropy = 0.993\nsamples =
1062\nvalue = [583, 479]'),
 Text(237.15, 54.359999999999985, 'X[9] <= 3.5\nentropy = 0.91\nsamples =
665\nvalue = [449, 216]'),
```

```
 Text(230.175, 18.119999999999976, 'entropy = 0.994\nsamples = 306\nvalue =
[167, 139]'),
 Text(244.12500000000003, 18.119999999999976, 'entropy = 0.75\nsamples =
359\nvalue = [282, 77]'),
 Text(265.05, 54.359999999999985, 'X[4] <= 1794.0\nentropy = 0.922\nsamples =
397\nvalue = [134, 263]'),
 Text(258.07500000000005, 18.119999999999976, 'entropy = 0.947\nsamples =
364\nvalue = [133, 231]'),
 Text(272.02500000000003, 18.119999999999976, 'entropy = 0.196\nsamples =
33\nvalue = [1, 32]'),
 Text(313.875, 126.83999999999999, 'X[2] <= 10.5\nentropy = 0.188\nsamples =
313\nvalue = [9, 304]'),
 Text(306.90000000000003, 90.6, 'X[5] <= 35.5\nentropy = 0.431\nsamples =
102\nvalue = [9, 93]'),
 Text(292.95000000000005, 54.359999999999985, 'X[0] <= 30.0\nentropy =
0.881\nsamples = 20\nvalue = [6, 14]'),
 Text(285.975, 18.119999999999976, 'entropy = 0.0\nsamples = 3\nvalue = [3,
0]'),
 Text(299.925, 18.119999999999976, 'entropy = 0.672\nsamples = 17\nvalue = [3,
14]'),
 Text(320.85, 54.359999999999985, 'X[1] <= 33379.0\nentropy = 0.226\nsamples =
82\nvalue = [3, 79]'),
 Text(313.875, 18.119999999999976, 'entropy = 1.0\nsamples = 4\nvalue = [2,
2]'),
 Text(327.82500000000005, 18.119999999999976, 'entropy = 0.099\nsamples =
78\nvalue = [1, 77]'),
 Text(320.85, 90.6, 'entropy = 0.0\nsamples = 211\nvalue = [0, 211]')]
```



[34]:
```
y_pred=dt.predict(X_test)
print("Accuracy for testing:",metrics.accuracy_score(y_test, y_pred))
ytrain_pred=dt.predict(X_train)
print("Accuracy for training:",metrics.accuracy_score(y_train, ytrain_pred))
print(metrics.classification_report(y_test,y_pred))
```

```
Accuracy for testing: 0.8403779214321233
Accuracy for training: 0.8443781341953666
              precision    recall  f1-score   support

           0       0.85      0.95      0.90      4533
           1       0.77      0.51      0.61      1500

    accuracy                           0.84      6033
   macro avg       0.81      0.73      0.76      6033
weighted avg       0.83      0.84      0.83      6033
```