Chapter 6 - exercise 2: Mushroom

Cho dữ liệu mushroom trong tập tin mushrooms.csv chứa thông tin của các mẫu nấm, nấm ăn được và không ăn được. Yêu cầu: Áp dụng thuật toán decision tree để cho biết nấm ăn được hay nấm độc dựa trên các thông tin được cung cấp.

- Dữ liệu có thể tham khảo và download tại: https://www.kaggle.com/jnduli/decision-tree-classifier-for-mushroom-dataset/data

Data Infromation

Bộ dữ liệu chứa 23 thuộc tính. Thuộc tính "class" là class attribute: Attribute Information: (classes: edible=e, poisonous=p)

- cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
- cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s
- cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r,pink=p,purple=u,red=e,white=w,yellow=y
- bruises: bruises=t,no=f
- odor: almond=a,anise=l,creosote=c,fishy=y,foul=f,musty=m,none=n,pungent=p,spicy=s
- gill-attachment: attached=a,descending=d,free=f,notched=n
- gill-spacing: close=c,crowded=w,distant=d
- gill-size: broad=b,narrow=n
- gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g, green=r,orange=o,pink=p,purple=u,red=e,white=w,yellow=y
- stalk-shape: enlarging=e,tapering=t
- stalk-root: bulbous=b,club=c,cup=u,equal=e,rhizomorphs=z,rooted=r,missing=?
- stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s
- stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s
- stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y
- stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y
- veil-type: partial=p,universal=u
- veil-color: brown=n,orange=o,white=w,yellow=y
- ring-number: none=n,one=o,two=t
- ring-type: cobwebby=c,evanescent=e,flaring=f,large=l,none=n,pendant=p,sheathing=s,zone=z
- spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r,orange=o,purple=u,white=w,yellow=y
- population: abundant=a,clustered=c,numerous=n,scattered=s,several=v,solitary=y
- habitat: grasses=g,leaves=l,meadows=m,paths=p,urban=u,waste=w,woods=d

Yêu cầu:

- Đọc dữ liệu, tìm hiểu sơ bộ về dữ liệu. Chuẩn hóa dữ liệu nếu cần
- Tạo X_train, X_test, y_train, y_test từ dữ liệu chuẩn hóa với tỷ lệ dữ liệu test là 0.3
- Áp dụng Decision Tree, Tìm kết quả
- Kiểm tra độ chính xác
- Trực quan hóa Decision Tree
- Đánh giá mô hình.
- Ghi mô hình nếu mô hình phù hợp

In [ ]:
```
# from google.colab import drive
# drive.mount("/content/gdrive", force_remount=True)
```

In [ ]:
```
#%cd '/content/gdrive/My Drive/LDS6_MachineLearning/practice/Chapter6_Decision_Tree/'
```

In [ ]:
```
import pandas as pd
import numpy as numpy
```

In [ ]:
```
dataset = pd.read_csv('mushrooms.csv', sep=",")
print(dataset.shape)
#dataset.info()
```

```
(8124, 23)
```

In [ ]:
```
dataset.head()
```

Out[ ]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color
0	p	x	s	n	t	p	f	c	n	k	...	s	w	w	p	w	o	p	k
1	e	x	s	y	t	a	f	c	b	k	...	s	w	w	p	w	o	p	n
2	e	b	s	w	t	l	f	c	b	n	...	s	w	w	p	w	o	p	n
3	p	x	y	w	t	p	f	c	n	n	...	s	w	w	p	w	o	p	k
4	e	x	s	g	f	n	f	w	b	k	...	s	w	w	p	w	o	e	n

5 rows × 23 columns

In [ ]:
```
# Vì các biến phân loại không tồn tại mối quan hệ thứ tự
# => cần chuẩn hóa bằng one hot encoder
```

In [ ]:
```
y = dataset['class']
x = dataset.drop(['class'], axis=1)
x = pd.get_dummies(x)
```

In [ ]:
```
x.head()
```

Out[ ]:

	cap-shape_b	cap-shape_c	cap-shape_f	cap-shape_k	cap-shape_s	cap-shape_x	cap-surface_f	cap-surface_g	cap-surface_s	cap-surface_y	...	population_s	population_v	population_y	ha
0	0	0	0	0	0	1	0	0	1	0	...	1	0	0	
1	0	0	0	0	0	1	0	0	1	0	...	0	0	0	
2	1	0	0	0	0	0	0	0	1	0	...	0	0	0	
3	0	0	0	0	0	1	0	0	0	1	...	1	0	0	
4	0	0	0	0	0	1	0	0	1	0	...	0	0	0	

5 rows × 117 columns

In [ ]:
```
y.head()
```

Out[ ]:
```
0    p
1    e
2    e
3    p
4    e
Name: class, dtype: object
```

In [ ]:
```
# trong trường hợp có quá nhiều cột dữ liệu có thể dùng dummy encoder để tạo các cột cần thiết mà không trùng lắp
features= pd.get_dummies(x, drop_first=True)
target = y
```

In [ ]:
```
features.head()
```

Out[ ]:

	cap-shape_b	cap-shape_c	cap-shape_f	cap-shape_k	cap-shape_s	cap-shape_x	cap-surface_f	cap-surface_g	cap-surface_s	cap-surface_y	...	population_s	population_v	population_y	ha
0	0	0	0	0	0	1	0	0	1	0	...	1	0	0	
1	0	0	0	0	0	1	0	0	1	0	...	0	0	0	
2	1	0	0	0	0	0	0	0	1	0	...	0	0	0	
3	0	0	0	0	0	1	0	0	0	1	...	1	0	0	
4	0	0	0	0	0	1	0	0	1	0	...	0	0	0	

5 rows × 117 columns

In [ ]:
```
# Đếm theo loại
occ = target.value_counts()
occ
```

Out[ ]:
```
e    4208
p    3916
Name: class, dtype: int64
```

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(features, target,
                                                    test_size=0.3,
                                                    random_state = 42)
```

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.utils.validation import column_or_1d
```

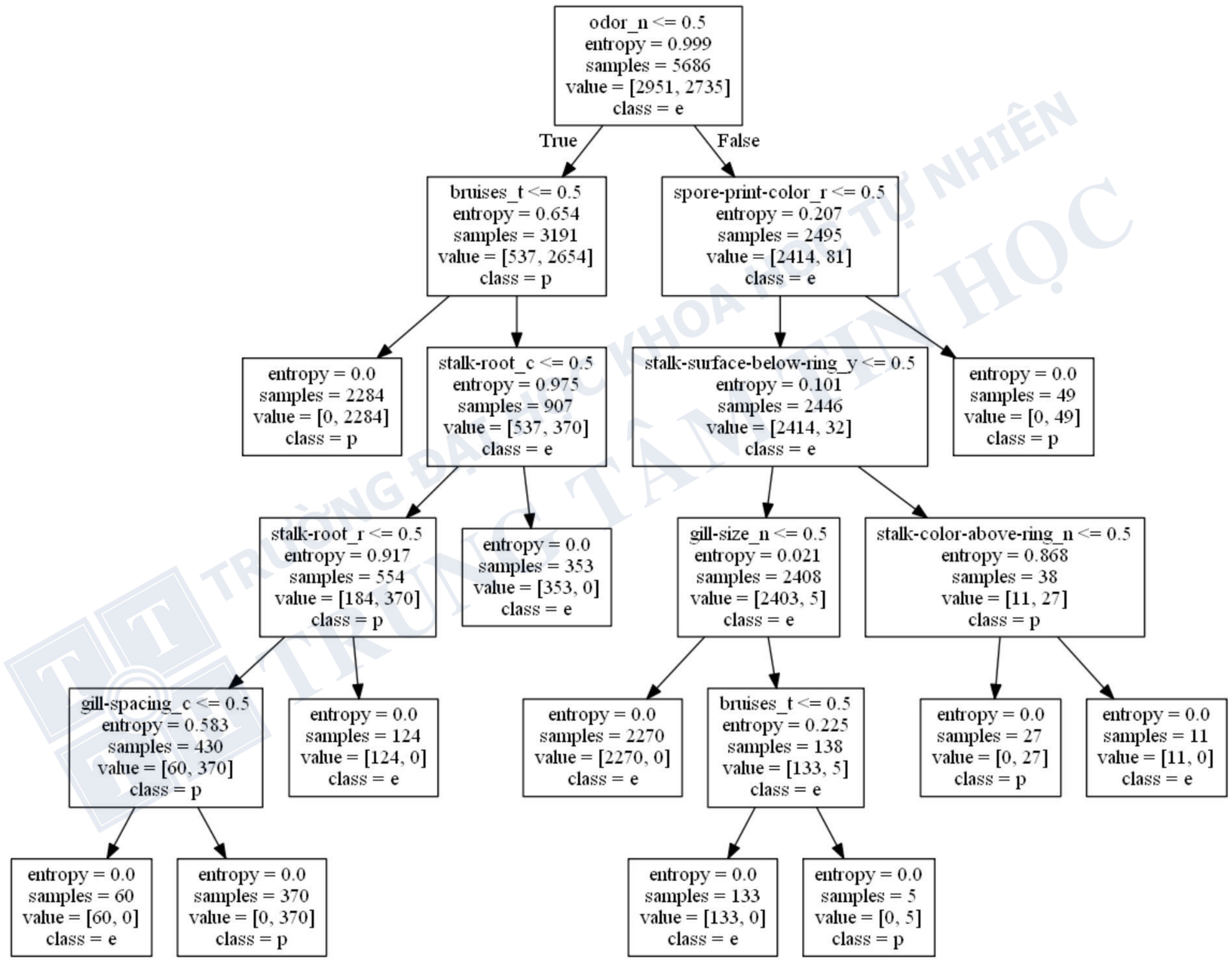
In [ ]:
```
tree_n = DecisionTreeClassifier(criterion= 'entropy') # criterion= 'entropy'
tree_n.fit(X_train,y_train)
```

Out[ ]:
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

In [ ]:
```
from IPython.display import Image
from sklearn import tree
import pydotplus
```

In [ ]:
```
dot_data = tree.export_graphviz(tree_n, out_file=None,
                                feature_names=features.columns,
                                class_names=['e', 'p'])
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

Out[ ]:



In [ ]:
```
# Kiểm tra độ chính xác
print("The Training prediction accuracy is:",
      tree_n.score(X_train,y_train)*100,"%")
print("The Testing prediction accuracy is:",
      tree_n.score(X_test,y_test)*100,"%")
```

```
The Training prediction accuracy is: 100.0 %
The Testing prediction accuracy is: 100.0 %
```

In [ ]:
```
# Đánh giá model
```

In [ ]:
```
y_pred = tree_n.predict(X_test)
```

In [ ]:
```
yTrain_pred = tree_n.predict(X_train)
```

In [ ]:
```
# Xem kết quả thống kê
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[1257    0]
 [   0 1181]]
              precision    recall  f1-score   support

           e       1.00      1.00      1.00      1257
           p       1.00      1.00      1.00      1181

    accuracy                           1.00      2438
   macro avg       1.00      1.00      1.00      2438
weighted avg       1.00      1.00      1.00      2438
```

Nhận xét:

- Cả train và test đều có Score cao, không bị overfitting/underfitting
- => Mdoel phù hợp

Apply PipeLine

In [ ]:
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
```

In [ ]:
```
#--cv1---cv2---cv3----cv4-... => result
```

In [ ]:
```
# 1, Doc du lieu
dataset1 = pd.read_csv('mushrooms.csv', sep=",")
```

In [ ]:
```
# 2, Xac dinh inputs/output => dua inputs/output vao X, y
y = dataset1['class']
X = dataset1.drop(['class'], axis=1) # gia tri trong cac cot la ky tu
```

In [ ]:
```
X_train_p, X_test_p, y_train_p, y_test_p = train_test_split(X, y,
                                                            test_size=0.3,
                                                            random_state = 42)
```

In [ ]:
```
# liet ke cac cong viec
# one_hot = OneHotEncoder(drop='first')
```

In [ ]:
```
# tree_p = DecisionTreeClassifier(criterion= 'entropy')
```

In [ ]:
```
pipe = Pipeline([('one_hot', OneHotEncoder(drop='first')),
                 #('scaler', StandardScaler()), ...
                 ('tree_p', DecisionTreeClassifier(criterion= 'entropy'))])
```

In [ ]:
```
pipe.fit(X_train_p, y_train_p)
```

Out[ ]:
```
Pipeline(memory=None,
         steps=[('one_hot',
                 OneHotEncoder(categorical_features=None, categories=None,
                               drop='first', dtype=<class 'numpy.float64'>,
                               handle_unknown='error', n_values=None,
                               sparse=True)),
                ('tree_p',
                 DecisionTreeClassifier(class_weight=None, criterion='entropy',
                                        max_depth=None, max_features=None,
                                        max_leaf_nodes=None,
                                        min_impurity_decrease=0.0,
                                        min_impurity_split=None,
                                        min_samples_leaf=1, min_samples_split=2,
                                        min_weight_fraction_leaf=0.0,
                                        presort=False, random_state=None,
                                        splitter='best'))],
         verbose=False)
```

In [ ]:
```
pipe.score(X_test_p, y_test_p)
```

Out[ ]:
```
1.0
```

In [ ]:
```
pipe.score(X_train_p, y_train_p)
```

Out[ ]:
```
1.0
```

In [ ]:
```
# model có kết quả tốt, không bị underfitting, overfitting
```

In [ ]:
```
y_hat_p = pipe.predict(X_test_p)
```

In [ ]:
```
# Xem kết quả thống kê
print(confusion_matrix(y_test_p, y_hat_p))
print(classification_report(y_test_p, y_hat_p))
```

```
[[1257    0]
 [   0 1181]]
              precision    recall  f1-score   support

           e       1.00      1.00      1.00      1257
           p       1.00      1.00      1.00      1181

    accuracy                           1.00      2438
   macro avg       1.00      1.00      1.00      2438
weighted avg       1.00      1.00      1.00      2438
```

In [ ]:
```
# model có độ chính xác cao
```

In [ ]:
```
pipe['one_hot']
```

Out[ ]:
```
OneHotEncoder(categorical_features=None, categories=None, drop='first',
              dtype=<class 'numpy.float64'>, handle_unknown='error',
              n_values=None, sparse=True)
```

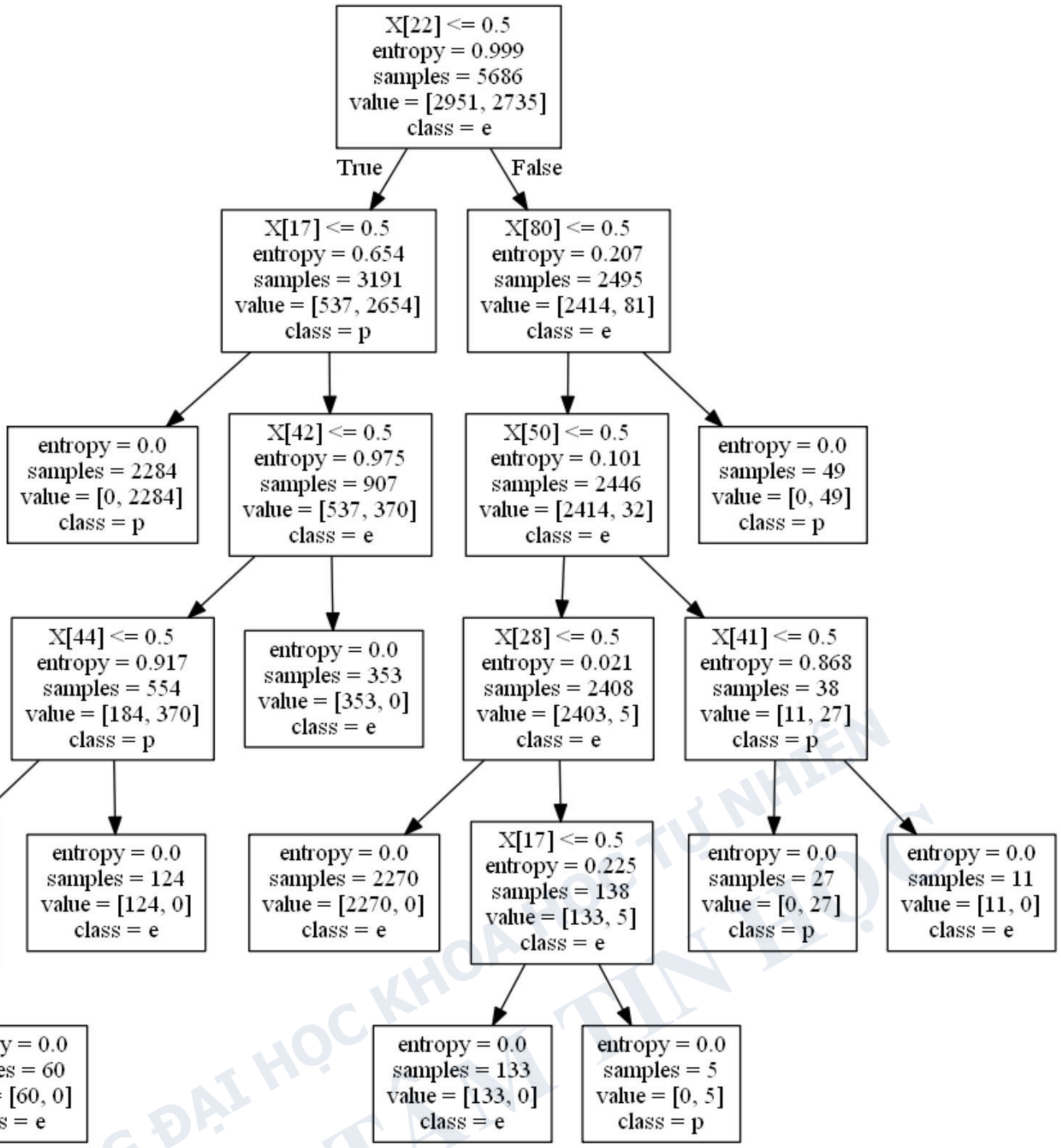
In [ ]:
```
pipe['tree_p']
```

Out[ ]:
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

In [ ]:
```
from IPython.display import Image
from sklearn import tree
import pydotplus
```

In [ ]:
```
dot_data_1 = tree.export_graphviz(pipe['tree_p'], out_file=None,
                                  class_names=['e', 'p'])
graph_1 = pydotplus.graph_from_dot_data(dot_data_1)
Image(graph_1.create_png())
```

Out[ ]:



In [ ]: