CS156 (Introduction to AI), Spring 2021

Homework Assignment #5 submission

Student Name: HUY NGUYEN

Student ID: 015207465

Email address: huy.l.nguyen@sjsu.edu (mailto:huy.l.nguyen@sjsu.edu)

Solution

In [1]:
```
import numpy as np
import pandas as pd
from sklearn import datasets
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import plot_confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn import tree
from sklearn.preprocessing import StandardScaler
```

In [2]:
```
np.random.seed(42)
```

In [3]:
```
dataset = pd.read_csv("homework5_input_data.csv")
dataset.head()
```

Out[3]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	st co abo r
0	p	x	s	n	t	p	f	c	n	k	...	s	
1	e	x	s	y	t	a	f	c	b	k	...	s	
2	e	b	s	w	t	l	f	c	b	n	...	s	
3	p	x	y	w	t	p	f	c	n	n	...	s	
4	e	x	s	g	f	n	f	w	b	k	...	s	

5 rows × 23 columns

In [4]:
```
X = dataset.drop(['class'], axis =1)
X.head()
```

Out[4]:

	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	...	stalk-surface-below-ring	st co ab
0	x	s	n	t	p	f	c	n	k	e	...	s	
1	x	s	y	t	a	f	c	b	k	e	...	s	
2	b	s	w	t	l	f	c	b	n	e	...	s	
3	x	y	w	t	p	f	c	n	n	e	...	s	
4	x	s	g	f	n	f	w	b	k	t	...	s	

5 rows × 22 columns

In [5]:
```
X_numeric = pd.get_dummies(X, columns=X.columns, prefix = X.columns)
X_numeric.head()
```

Out[5]:

	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	surfa
0	0	0	0	0	0	1	0	0	1	
1	0	0	0	0	0	1	0	0	1	
2	1	0	0	0	0	0	0	0	1	
3	0	0	0	0	0	1	0	0	0	
4	0	0	0	0	0	1	0	0	1	

5 rows × 117 columns

In [6]:
```
Y = pd.DataFrame(dataset['class'])
Y
```

Out[6]:

	class
0	p
1	e
2	e
3	p
4	e
...	...
8119	e
8120	e
8121	e
8122	p
8123	e

8124 rows × 1 columns

In [7]:
```
drop_nums = {"class": {"e": 0, "p": 1}}
Y_numeric = Y.replace(drop_nums)
FinalData =  pd.concat([Y_numeric, X_numeric], axis = 1)
FinalData
```

Out[7]:

	class	cap-shape_b	cap-shape_c	cap-shape_f	cap-shape_k	cap-shape_s	cap-shape_x	cap-surface_f	cap-surface_g	surfac
0	1	0	0	0	0	0	1	0	0	
1	0	0	0	0	0	0	1	0	0	
2	0	1	0	0	0	0	0	0	0	
3	1	0	0	0	0	0	1	0	0	
4	0	0	0	0	0	0	1	0	0	
...	...	...	...	...	...	...	...	...	...	
8119	0	0	0	0	1	0	0	0	0	
8120	0	0	0	0	0	0	1	0	0	
8121	0	0	0	1	0	0	0	0	0	
8122	1	0	0	0	1	0	0	0	0	
8123	0	0	0	0	0	0	1	0	0	

8124 rows × 118 columns

In [8]:
```
Y_final = FinalData['class']
Y_final
```

Out[8]:
```
0       1
1       0
2       0
3       1
4       0
       ..
8119    0
8120    0
8121    0
8122    1
8123    0
Name: class, Length: 8124, dtype: int64
```

In [9]:
```
X = FinalData.drop(['class'], axis =1).values
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

In [10]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y_final, test_size=0.2, ra
X_train.shape, Y_train.shape, X_test.shape, Y_test.shape
```

Out[10]:
```
((6499, 117), (6499,), (1625, 117), (1625,))
```

In [11]:
```
X_train
```

Out[11]:
```
array([[-0.24272523, -0.02219484, -0.79620985, ..., -0.40484176,
        -0.21782364, -0.15558197],
       [-0.24272523, -0.02219484, -0.79620985, ..., -0.40484176,
        -0.21782364, -0.15558197],
       [-0.24272523, -0.02219484, -0.79620985, ..., -0.40484176,
        -0.21782364, -0.15558197],
       ...,
       [-0.24272523, -0.02219484, -0.79620985, ..., -0.40484176,
        -0.21782364, -0.15558197],
       [-0.24272523, -0.02219484,  1.2559503 , ..., -0.40484176,
        -0.21782364, -0.15558197],
       [-0.24272523, -0.02219484, -0.79620985, ..., -0.40484176,
        -0.21782364, -0.15558197]])
```

In [12]:
```
Y_train
```

Out[12]:
```
7434    0
7725    0
783     0
1928    0
7466    1
       ..
4931    0
3264    1
1653    0
2607    0
2732    0
Name: class, Length: 6499, dtype: int64
```

In [13]:
```
X_test
```

Out[13]:
```
array([[-0.24272523, -0.02219484, -0.79620985, ..., -0.40484176,
         4.59086996, -0.15558197],
       [-0.24272523, -0.02219484,  1.2559503 , ..., -0.40484176,
        -0.21782364, -0.15558197],
       [-0.24272523, -0.02219484, -0.79620985, ..., -0.40484176,
        -0.21782364, -0.15558197],
       ...,
       [-0.24272523, -0.02219484,  1.2559503 , ...,  2.47010093,
        -0.21782364, -0.15558197],
       [ 4.11988487, -0.02219484, -0.79620985, ..., -0.40484176,
        -0.21782364, -0.15558197],
       [-0.24272523, -0.02219484, -0.79620985, ..., -0.40484176,
        -0.21782364, -0.15558197]])
```

In [14]:
```
Y_test
```

Out[14]:
```
380     1
3641    0
273     0
1029    0
684     0
       ..
3535    0
1643    0
6494    1
6       0
3175    0
Name: class, Length: 1625, dtype: int64
```

Build a decision tree classifier

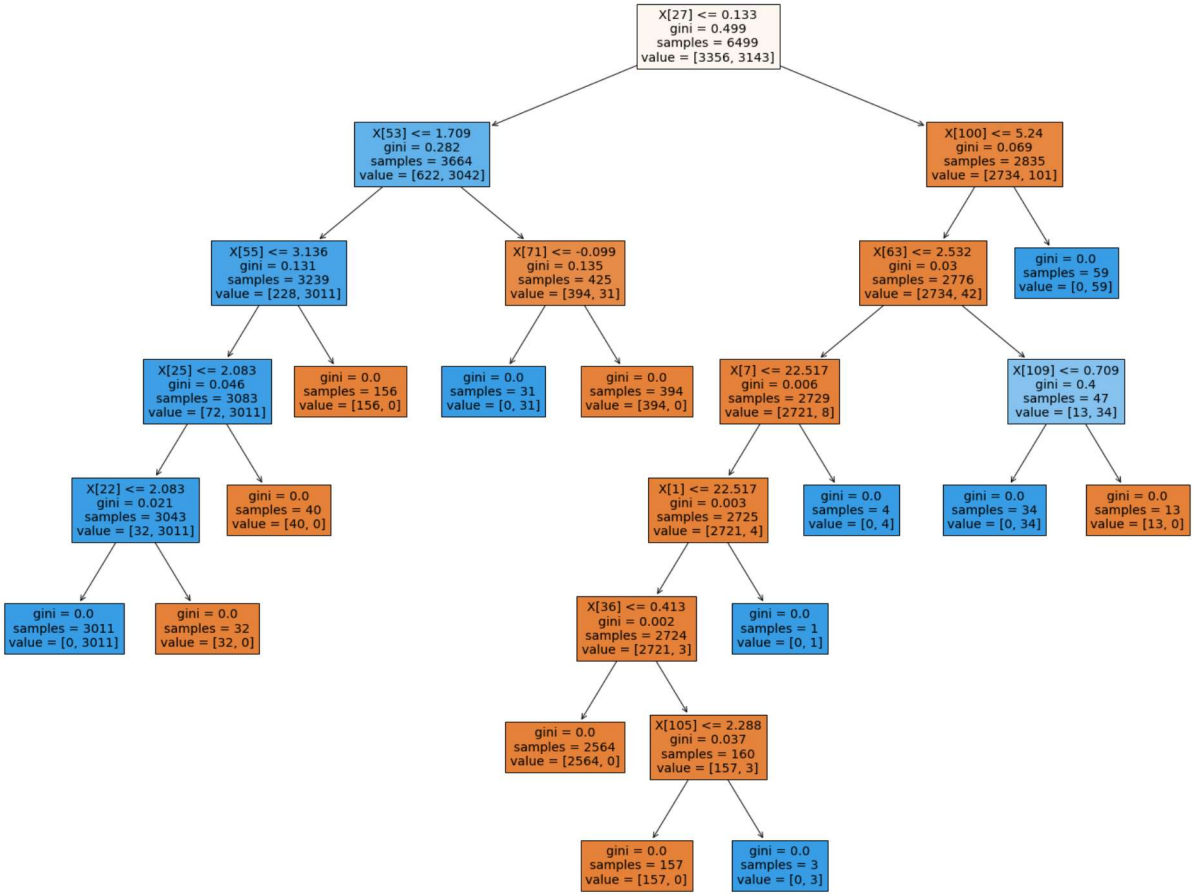
In [15]:
```
model = DecisionTreeClassifier(random_state=0)

# we can first score our model through cross validation (applicable to any superv
cross_val_score(model, X_train, Y_train, cv=5)
```

Out[15]:
```
array([1., 1., 1., 1., 1.])
```

In [16]:
```
model.fit(X_train, Y_train)

print('Accuracy of linear SVC on training set: {:.2f}'.format(model.score(X_trair

print('Accuracy of linear SVC on test set: {:.2f}'.format(model.score(X_test, Y_t
```

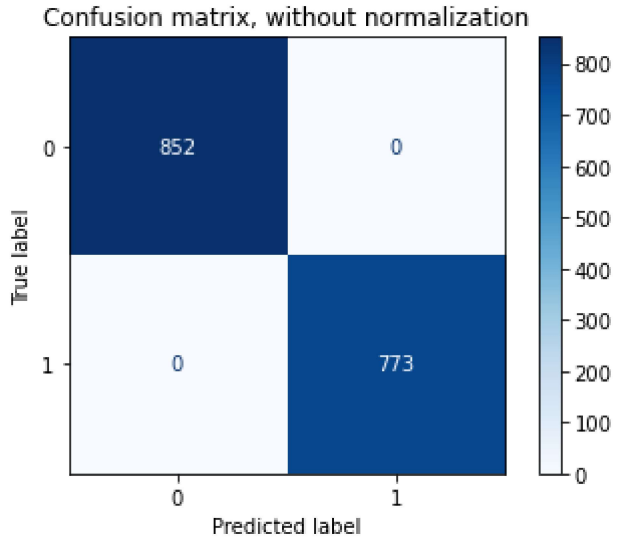
```
Accuracy of linear SVC on training set: 1.00
Accuracy of linear SVC on test set: 1.00
```

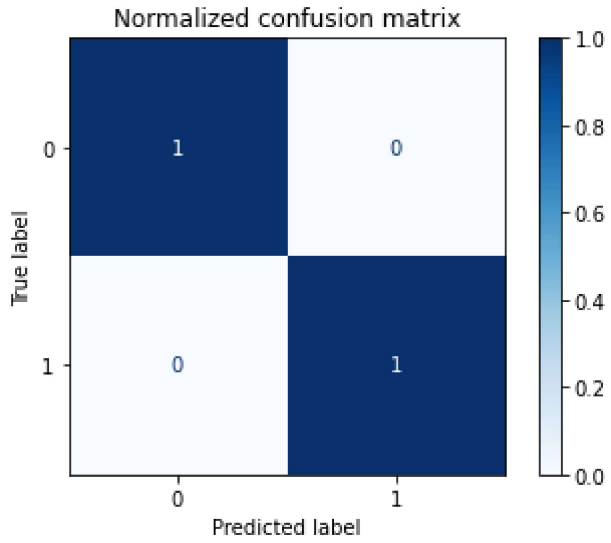
In [17]:
```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(model, filled=True)
```



In [18]:
```
# this code is adopted from this example:
# https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_ma

np.set_printoptions(precision=2)
titles_options = [("Confusion matrix, without normalization", None),
                  ("Normalized confusion matrix", 'true')]
for title, normalize in titles_options:
    disp = plot_confusion_matrix(model, X_test, Y_test,
                                 display_labels=np.unique(Y_final),
                                 cmap=plt.cm.Blues,
                                 normalize=normalize)
    disp.ax_.set_title(title)

    print(title)
    print(disp.confusion_matrix)

plt.show()
```

```
Confusion matrix, without normalization
[[852   0]
 [  0 773]]
Normalized confusion matrix
[[1. 0.]
 [0. 1.]]
```




Normalized confusion matrix
1
0
0
1
0
1
True label
Predicted label
1.0
0.8
0.6
0.4
0.2
0.0

In [ ]: