In [23]:

```
#LIBRARIES
import mglearn as mglearn
import os
import graphviz
import matplotlib.pyplot as plt
import sklearn.datasets as datasets
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from IPython.display import Image
from sklearn.tree import export_graphviz
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_moons
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.datasets import make_blobs
from sklearn.svm import LinearSVC
import sklearn.svm as linear_svm
from mpl_toolkits.mplot3d import Axes3D, axes3d
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from tensorflow.examples.tutorials.mnist import input_data
from sklearn.datasets import make_circles
from sklearn.datasets import load_wine
from sklearn.datasets import fetch_california_housing
from sklearn.neural_network import MLPRegressor
```

In [2]:

```
#DECISION TREE
wine = load_wine()
print(wine.DESCR)
wine = datasets.load_wine()
df = pd.DataFrame(wine.data,columns=wine.feature_names)
df['target'] = pd.Series(wine.target)
df.columns = ['Class label', 'Alcohol', 'Malic acid', 'Ash',
              'Alcalinity of ash', 'Magnesium', 'Total phenols',
              'Flavanoids', 'Nonflavanoid phenols', 'Proanthocyanins',
              'Color intensity', 'Hue', 'OD280/OD315 of diluted wines', 'Proline
']
X = df.drop('Class label', 1)
y = df['Class label']
df.head()
```

```
Wine Data Database
```

==========

Notes

Data Set Characteristics:
    :Number of Instances: 178 (50 in each of three classes)
    :Number of Attributes: 13 numeric, predictive attributes and the
class
    :Attribute Information:
                - 1) Alcohol
                - 2) Malic acid
                - 3) Ash
                - 4) Alcalinity of ash
                - 5) Magnesium
                - 6) Total phenols
                - 7) Flavanoids
                - 8) Nonflavanoid phenols
                - 9) Proanthocyanins
                - 10)Color intensity
                - 11)Hue
                - 12)OD280/OD315 of diluted wines
                - 13)Proline
                - class:
                - class_0
                - class_1
                - class_2

:Summary Statistics:

	Min	Max	Mean	SD
Alcohol:	11.0	14.8	13.0	0.8
Malic Acid:	0.74	5.80	2.34	1.12
Ash:	1.36	3.23	2.36	0.27
Alcalinity of Ash:	10.6	30.0	19.5	3.3
Magnesium:	70.0	162.0	99.7	14.3
Total Phenols:	0.98	3.88	2.29	0.63
Flavanoids:	0.34	5.08	2.03	1.00
Nonflavanoid Phenols:	0.13	0.66	0.36	0.12
Proanthocyanins:	0.41	3.58	1.59	0.57
Colour Intensity:	1.3	13.0	5.1	2.3
Hue:	0.48	1.71	0.96	0.23
OD280/OD315 of diluted wines:	1.27	4.00	2.61	0.71
Proline:	278	1680	746	315

:Missing Attribute Values: None
:Class Distribution: class_0 (59), class_1 (71), class_2 (48)
:Creator: R.A. Fisher
:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
:Date: July, 1988

This is a copy of UCI ML Wine recognition datasets.
https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data

The data is the results of a chemical analysis of wines grown in the same
region in Italy by three different cultivators. There are thirteen different
measurements taken for different constituents found in the three types of
wine.

Original Owners:

Forina, M. et al, PARVUS -
An Extendible Package for Data Exploration, Classification and Correlation.
Institute of Pharmaceutical and Food Analysis and Technologies,
Via Brigata Salerno, 16147 Genoa, Italy.

Citation:

Lichman, M. (2013). UCI Machine Learning Repository
[http://archive.ics.uci.edu/ml]. Irvine, CA: University of California,
School of Information and Computer Science.

References
----
(1)
S. Aeberhard, D. Coomans and O. de Vel,
Comparison of Classifiers in High Dimensional Settings,
Tech. Rep. no. 92-02, (1992), Dept. of Computer Science and Dept. of
Mathematics and Statistics, James Cook University of North Queensland.
(Also submitted to Technometrics).

The data was used with many others for comparing various
classifiers. The classes are separable, though only RDA
has achieved 100% correct classification.
(RDA : 100%, QDA 99.4%, LDA 98.9%, 1NN 96.1% (z-transformed data))
(All results using the leave-one-out technique)

(2)
S. Aeberhard, D. Coomans and O. de Vel,
"THE CLASSIFICATION PERFORMANCE OF RDA"
Tech. Rep. no. 92-01, (1992), Dept. of Computer Science and Dept. of
Mathematics and Statistics, James Cook University of North Queensland.
(Also submitted to Journal of Chemometrics).

Out[2]:

	Class label	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoi phenol
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82

In [3]:

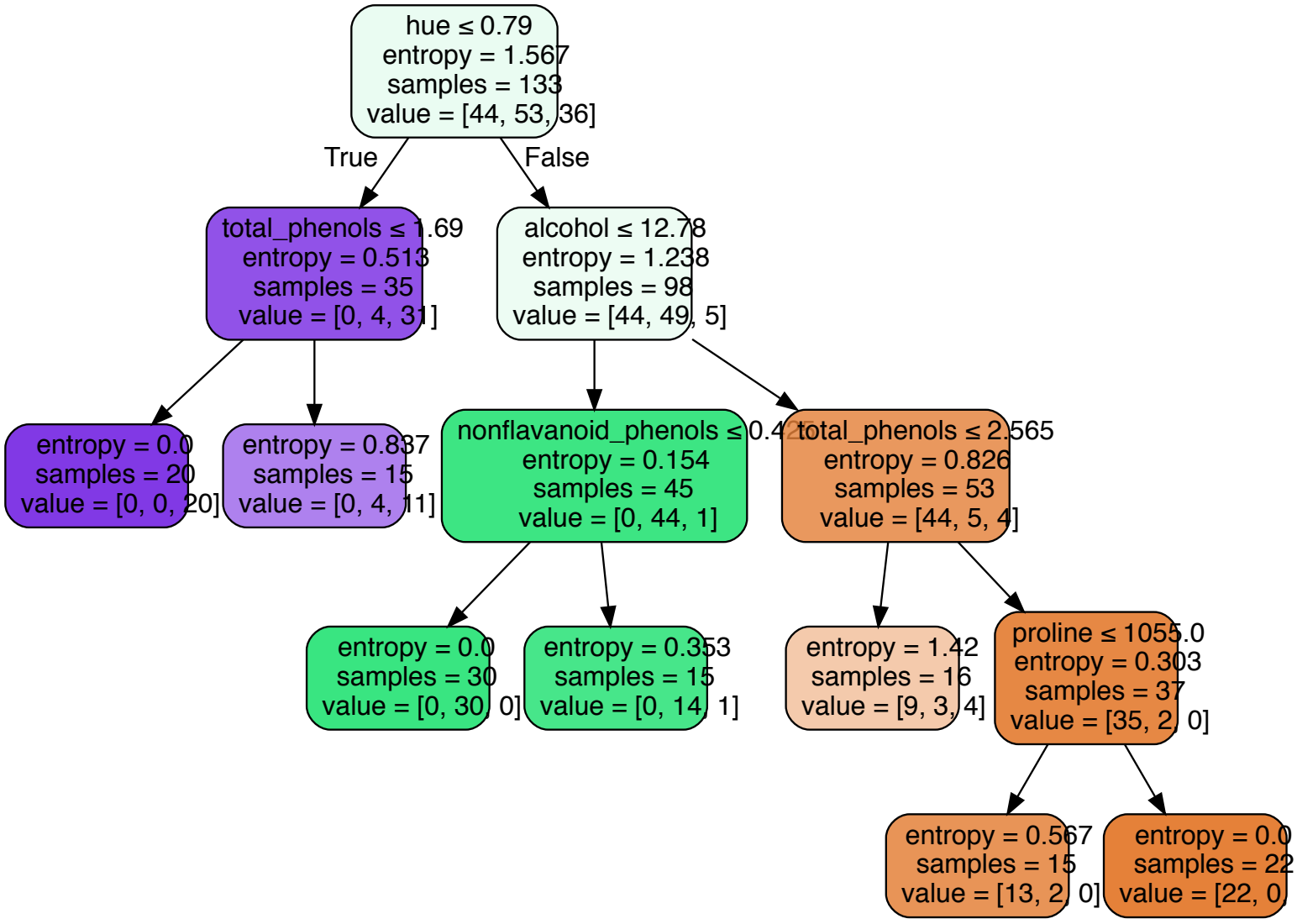
```
X_train, X_test, y_train, y_test = train_test_split(wine.data,
                                                    wine.target,
                                                    stratify = wine.target,
                                                    random_state = 42)
tree = DecisionTreeClassifier(criterion = 'entropy',
                              min_samples_split = 20,
                              min_samples_leaf = 15,
                              max_features = 'sqrt',
                              max_leaf_nodes = 12,
                              random_state = 0)
tree.fit(X_train, y_train)
y_pred = tree.predict(X_test)
print('#Training data points: %d' % X_train.shape[0])
print('#Testing data points: %d' % X_test.shape[0])
print("")
print("Accuracy on training set: {:,.3f}".format(tree.score(X_train, y_train)))
print("Accuracy on test set: {:,.3f}".format(tree.score(X_test, y_test)))
print("")
print('Class labels:', np.unique(wine.target))
print('Misclassified samples: %d' % (y_test != y_pred).sum())
```

```
#Training data points: 133
#Testing data points: 45

Accuracy on training set: 0.895
Accuracy on test set: 0.844

Class labels: [0 1 2]
Misclassified samples: 7
```

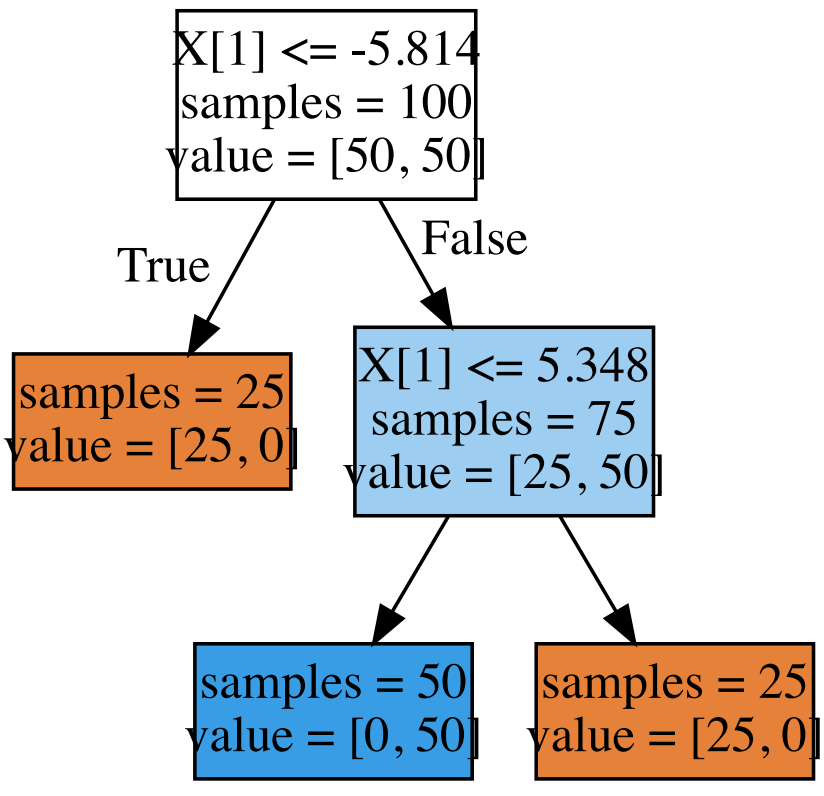
In [4]:

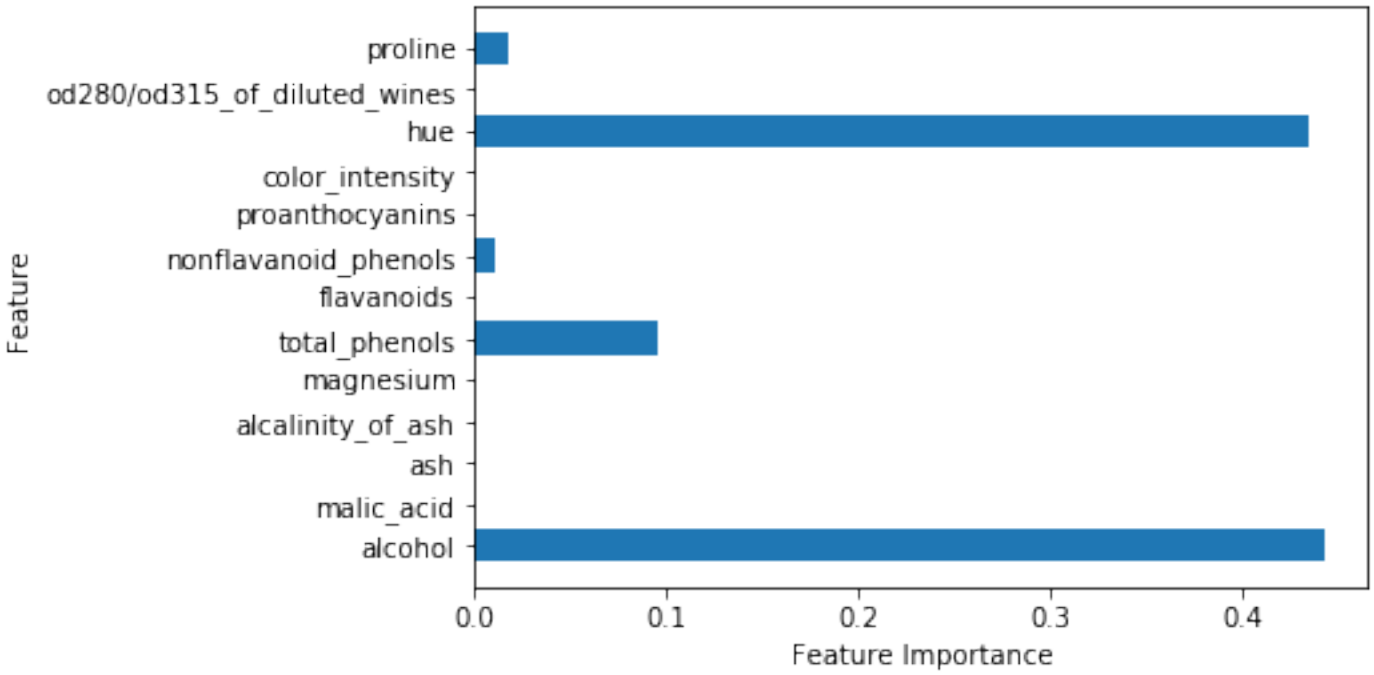
```
X_train, X_test, y_train, y_test = train_test_split(wine.data,
                                                    wine.target,
                                                    stratify = wine.target,
                                                    random_state = 42)

tree = DecisionTreeClassifier(criterion = 'entropy',
                              min_samples_split = 20,
                              min_samples_leaf = 15,
                              max_features = 'sqrt',
                              max_leaf_nodes = 12,
                              random_state = 0)
tree.fit(X_train, y_train)

export_graphviz(tree, out_file = "tree.dot",
                feature_names = wine.feature_names,
                filled = True, rounded = True,
               special_characters = True) #[writes data into .dot file]

with open("tree.dot") as f:
    dot_graph = f.read()
display(graphviz.Source(dot_graph))

print("Feature importance: \n{}".format(tree.feature_importances_))
```

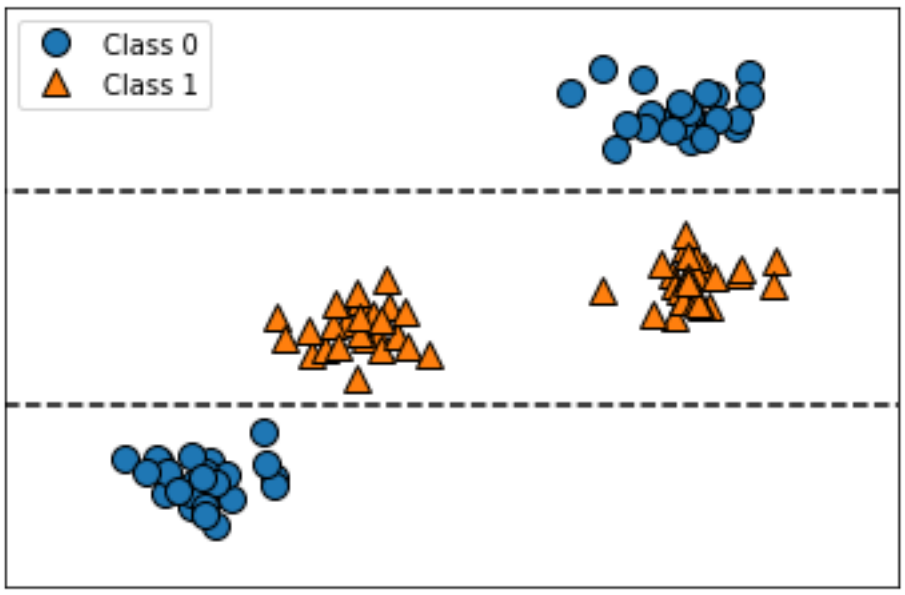
```
def plot_feature_importances_wine(model):
    n_features = wine.data.shape[1]
    plt.barh(range(n_features), model.feature_importances_, align = 'center')
    plt.yticks(np.arange(n_features), wine.feature_names)
    plt.xlabel("Feature Importance")
    plt.ylabel("Feature")
    plt.ylim(-1, n_features)
plot_feature_importances_wine(tree)

tree = mglearn.plots.plot_tree_not_monotone()

display(tree)
```




```
Feature importance:
[0.44311648 0.         0.         0.         0.         0.09543221
 0.         0.01015208 0.         0.         0.43418745 0.
 0.01711179]
Feature importances: [0. 1.]
```



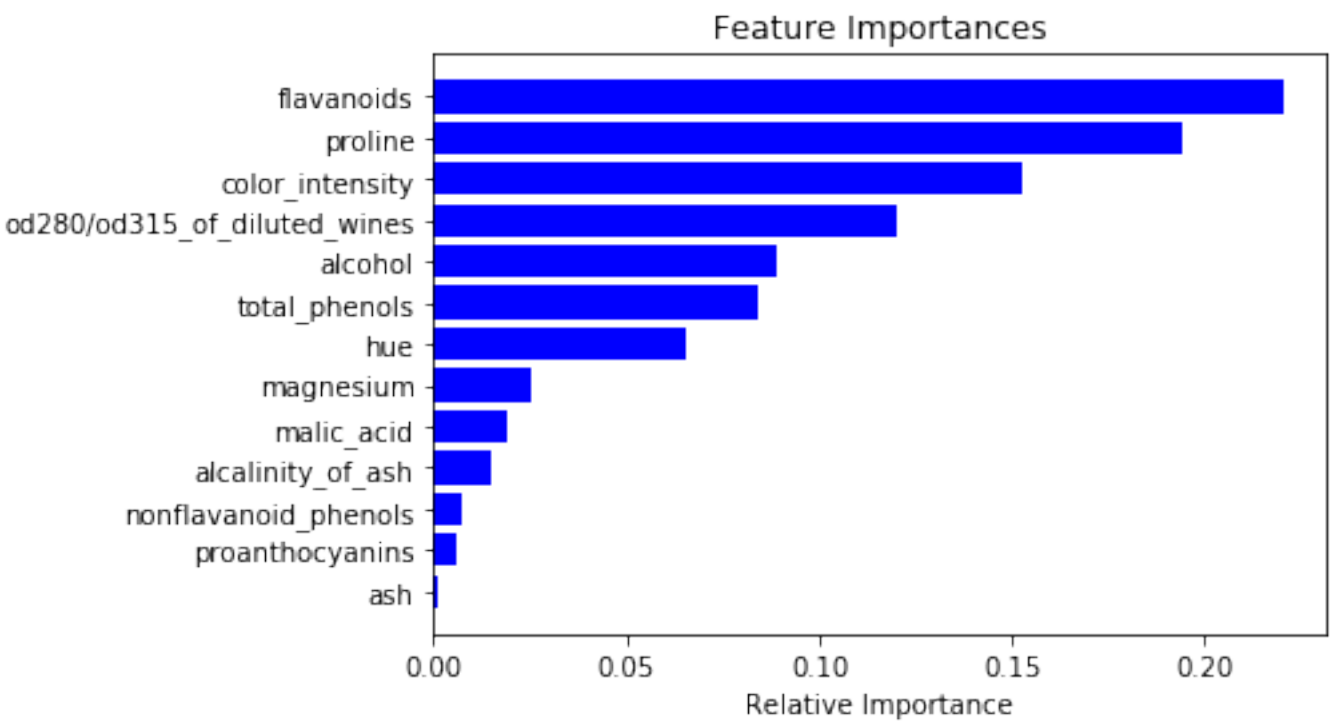

proline
od280/od315_of_diluted_wines
hue
color_intensity
proanthocyanins
nonflavanoid_phenols
flavanoids
total_phenols
magnesium
alcalinity_of_ash
ash
malic_acid
alcohol
Feature
0.0
0.1
0.2
0.3
0.4
Feature Importance


Class 0
Class 1

In [15]:

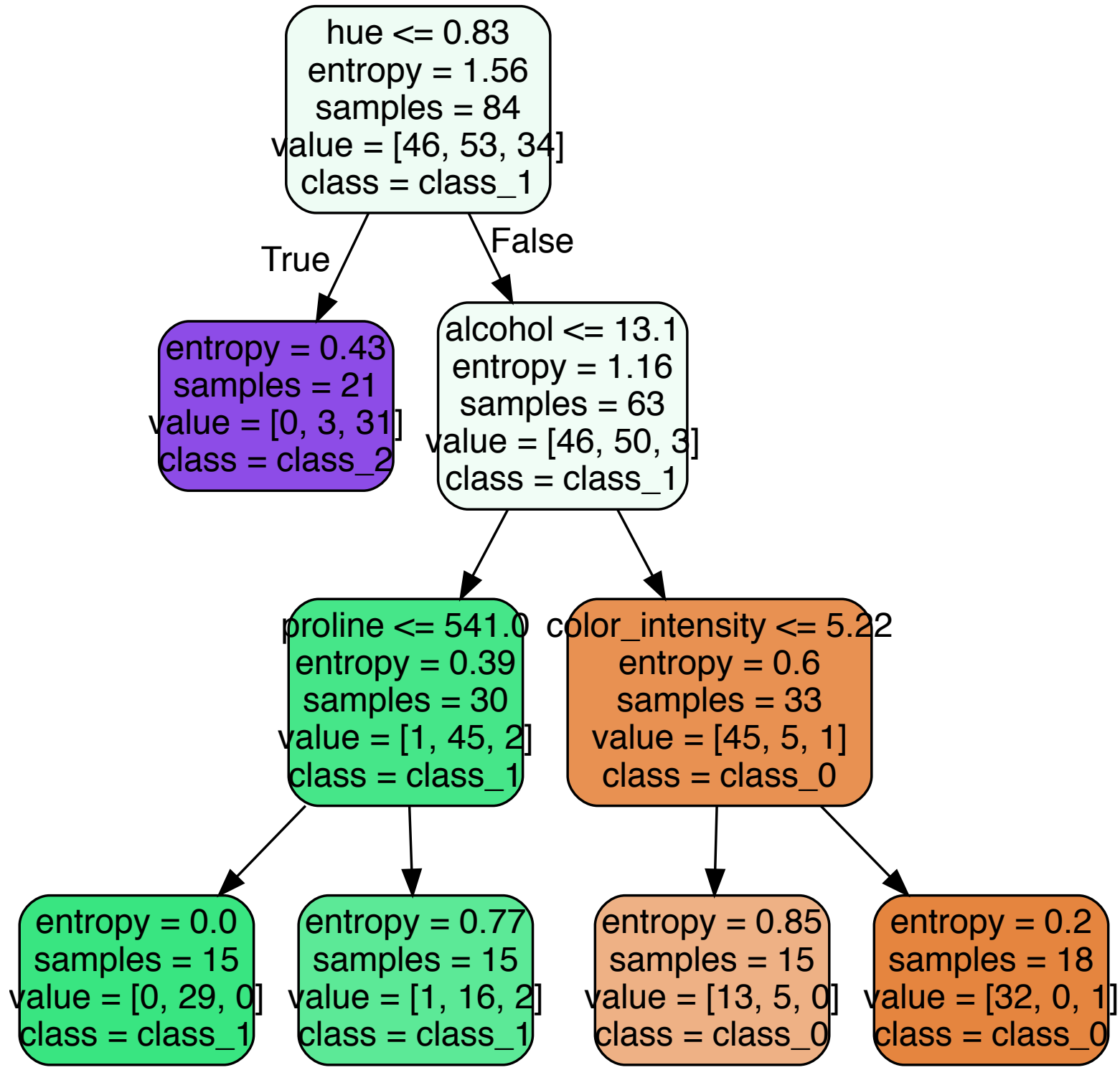
```
#RANDOM FOREST
forest = RandomForestClassifier(criterion='entropy',
                                n_estimators = 50,
                                 min_samples_split = 20,
                                min_samples_leaf = 15,
                                max_features = 'sqrt',
                                max_leaf_nodes = 12,
                                random_state = 0)
forest.fit(X_train, y_train)

y_pred = forest.predict(X_test)
print("Accuracy on training set: {:.3f}".format(forest.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(forest.score(X_test, y_test)))


importances = forest.feature_importances_
features = wine['feature_names']
indices = np.argsort(importances)



plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='b', align='center')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```

```
Accuracy on training set: 0.985
Accuracy on test set: 1.000
```



In [6]:

```
estimator = forest.estimators_[29]

export_graphviz(estimator, out_file = "rftree.dot",
                feature_names = wine.feature_names,
                class_names = wine.target_names,
                rounded = True, proportion = False,
                precision = 2, filled = True)


with open("rftree.dot") as f:
    dot_graph = f.read()
display(graphviz.Source(dot_graph))
```



In [16]:

```
importances = forest.feature_importances_

#sorted indices
indices = np.argsort(importances)[::-1]

for f in range(X_train.shape[1]):
    print("%2d) %-*s %f" % (f + 1, 30,
                            X.columns.values[indices[f]],
                            importances[indices[f]]))
```

```
 1) Flavanoids                     0.220670
 2) Proline                        0.194247
 3) Color intensity                0.152337
 4) OD280/OD315 of diluted wines   0.120117
 5) Alcohol                        0.088855
 6) Total phenols                  0.084423
 7) Hue                            0.065712
 8) Magnesium                      0.025427
 9) Malic acid                     0.018752
10) Alcalinity of ash              0.015239
11) Nonflavanoid phenols           0.007226
12) Proanthocyanins                0.005836
13) Ash                            0.001158
```

In [21]:

```
#Gradient Boosted Trees MaxDepth = 0.01
X_train, X_test, y_train, y_test = train_test_split(wine.data, wine.target, rand
om_state=0)

gbrt=GradientBoostingClassifier(random_state=0, learning_rate=0.01)

gbrt.fit(X_train, y_train)
print("Accuracy on training set: {:.3f}".format(gbrt.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(gbrt.score(X_test, y_test)))

importances = gbrt.feature_importances_
features = wine['feature_names']
indices = np.argsort(importances)

def plot_feature_importances_wine(model):
       n_features = wine.data.shape[1]
       plt.barh(range(n_features), model.feature_importances_, align='center')
       plt.yticks(np.arange(n_features), wine.feature_names)
       plt.xlabel("Feature importance")
       plt.ylabel("Fetaure")
       plt.ylim(-1, n_features)
       plot_feature_importances_wine(gbrt)
```

```
Accuracy on training set: 1.000
Accuracy on test set: 0.933
```

In [9]:

```
#Gradient Boosted Trees MaxDepth = 0.05
X_train, X_test, y_train, y_test = train_test_split(wine.data, wine.target, rand
om_state=0)

gbrt=GradientBoostingClassifier(random_state=42, learning_rate=0.05)

gbrt.fit(X_train, y_train)
print("Accuracy on training set: {:.3f}".format(gbrt.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(gbrt.score(X_test, y_test)))

importances = gbrt.feature_importances_
features = wine['feature_names']
indices = np.argsort(importances)

def plot_feature_importances_wine(model):
       n_features = wine.data.shape[1]
       plt.barh(range(n_features), model.feature_importances_, align='center')
       plt.yticks(np.arange(n_features), wine.feature_names)
       plt.xlabel("Feature importance")
       plt.ylabel("Fetaure")
       plt.ylim(-1, n_features)
       plot_feature_importances_wine(gbrt)
```

```
Accuracy on training set: 1.000
Accuracy on test set: 0.956
```

In [10]:

```
#Gradient Boosted Trees MaxDepth = 0.1
X_train, X_test, y_train, y_test = train_test_split(wine.data, wine.target, rand
om_state=0)

gbrt=GradientBoostingClassifier(random_state=42, learning_rate=0.1)

gbrt.fit(X_train, y_train)
print("Accuracy on training set: {:.3f}".format(gbrt.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(gbrt.score(X_test, y_test)))

importances = gbrt.feature_importances_
features = wine['feature_names']
indices = np.argsort(importances)

def plot_feature_importances_wine(model):
       n_features = wine.data.shape[1]
       plt.barh(range(n_features), model.feature_importances_, align='center')
       plt.yticks(np.arange(n_features), wine.feature_names)
       plt.xlabel("Feature importance")
       plt.ylabel("Fetaure")
       plt.ylim(-1, n_features)
       plot_feature_importances_wine(gbrt)
```

```
Accuracy on training set: 1.000
Accuracy on test set: 0.956
```

In [11]:

```
#Gradient Boosted Trees MaxDepth = 1
X_train, X_test, y_train, y_test = train_test_split(wine.data,
                                                    wine.target,
                                                    stratify = wine.target,
                                                    random_state = 0)

gbrt=GradientBoostingClassifier(random_state=42, max_depth=1)

gbrt.fit(X_train, y_train)
print("Accuracy on training set: {:.3f}".format(gbrt.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(gbrt.score(X_test, y_test)))

importances = gbrt.feature_importances_
features = wine['feature_names']
indices = np.argsort(importances)

def plot_feature_importances_wine(model):
       n_features = wine.data.shape[1]
       plt.barh(range(n_features), model.feature_importances_, align='center')
       plt.yticks(np.arange(n_features), wine.feature_names)
       plt.xlabel("Feature importance")
       plt.ylabel("Fetaure")
       plt.ylim(-1, n_features)
       plot_feature_importances_wine(gbrt)
```

```
Accuracy on training set: 1.000
Accuracy on test set: 1.000
```

In [24]:

```
ch = fetch_california_housing()
df = pd.DataFrame(ch.data,columns=ch.feature_names)
df.head()
```

Out[24]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longi
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.2
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.2
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.2
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.2
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.2

In [26]:

```
X_train, X_test, y_train, y_test = train_test_split(ch.data, ch.target, random_s
tate=42)

mlp = MLPRegressor(hidden_layer_sizes = (20,),
                   activation = 'relu',
                   solver = 'adam',
                   learning_rate = 'adaptive',
                   max_iter = 2000,
                   learning_rate_init = 0.01,
                   alpha = 0.01,
                   random_state = 0).fit(X_train, y_train)
```

In [27]:

```
print("Adjusted R-square on training set: {:.3f}".format(mlp.score(X_train, y_tr
ain)))
print("Adjusted R-square on test set: {:.3f}".format(mlp.score(X_test, y_test)))
```

```
Adjusted R-square on training set: 0.466
Adjusted R-square on test set: 0.451
```