In [91]:

```
import sys
!{sys.executable} -m pip install nbconvert
```

```
Requirement already satisfied: nbconvert in /home/michael/anaconda3/
lib/python3.7/site-packages (5.6.0)
Requirement already satisfied: entrypoints>=0.2.2 in /home/michael/a
naconda3/lib/python3.7/site-packages (from nbconvert) (0.3)
Requirement already satisfied: jinja2>=2.4 in /home/michael/anaconda
3/lib/python3.7/site-packages (from nbconvert) (2.10.3)
Requirement already satisfied: pandocfilters>=1.4.1 in /home/michae
l/anaconda3/lib/python3.7/site-packages (from nbconvert) (1.4.2)
Requirement already satisfied: jupyter-core in /home/michael/anacond
a3/lib/python3.7/site-packages (from nbconvert) (4.5.0)
Requirement already satisfied: nbformat>=4.4 in /home/michael/anacon
da3/lib/python3.7/site-packages (from nbconvert) (4.4.0)
Requirement already satisfied: pygments in /home/michael/anaconda3/l
ib/python3.7/site-packages (from nbconvert) (2.4.2)
Requirement already satisfied: mistune<2,>=0.8.1 in /home/michael/an
aconda3/lib/python3.7/site-packages (from nbconvert) (0.8.4)
Requirement already satisfied: traitlets>=4.2 in /home/michael/anaco
nda3/lib/python3.7/site-packages (from nbconvert) (4.3.3)
Requirement already satisfied: defusedxml in /home/michael/anaconda
3/lib/python3.7/site-packages (from nbconvert) (0.6.0)
Requirement already satisfied: bleach in /home/michael/anaconda3/li
b/python3.7/site-packages (from nbconvert) (3.1.0)
Requirement already satisfied: testpath in /home/michael/anaconda3/l
ib/python3.7/site-packages (from nbconvert) (0.4.2)
Requirement already satisfied: MarkupSafe>=0.23 in /home/michael/ana
conda3/lib/python3.7/site-packages (from jinja2>=2.4->nbconvert) (1.
1.1)
Requirement already satisfied: ipython-genutils in /home/michael/ana
conda3/lib/python3.7/site-packages (from nbformat>=4.4->nbconvert)
(0.2.0)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in /home/mich
ael/anaconda3/lib/python3.7/site-packages (from nbformat>=4.4->nbcon
vert) (3.0.2)
Requirement already satisfied: six in /home/michael/anaconda3/lib/py
thon3.7/site-packages (from traitlets>=4.2->nbconvert) (1.12.0)
Requirement already satisfied: decorator in /home/michael/anaconda3/
lib/python3.7/site-packages (from traitlets>=4.2->nbconvert) (4.4.0)
Requirement already satisfied: webencodings in /home/michael/anacond
a3/lib/python3.7/site-packages (from bleach->nbconvert) (0.5.1)
Requirement already satisfied: attrs>=17.4.0 in /home/michael/anacon
da3/lib/python3.7/site-packages (from jsonschema!=2.5.0,>=2.4->nbfor
mat>=4.4->nbconvert) (19.2.0)
Requirement already satisfied: setuptools in /home/michael/anaconda
3/lib/python3.7/site-packages (from jsonschema!=2.5.0,>=2.4->nbforma
t>=4.4->nbconvert) (41.4.0)
Requirement already satisfied: pyrsistent>=0.14.0 in /home/michael/a
naconda3/lib/python3.7/site-packages (from jsonschema!=2.5.0,>=2.4->
nbformat>=4.4->nbconvert) (0.15.4)
```

In [ ]:

```
# import os
# from sklearn.tree import export_graphviz
# import six
# import pydot
# from sklearn import tree
# dotfile = six.StringIO()
# i_tree = 0
# for tree_in_forest in estimator.estimators_:
#     export_graphviz(tree_in_forest,out_file='tree.dot',
#     feature_names=col,
#     filled=True,
#     rounded=True)
#     (graph,) = pydot.graph_from_dot_file('tree.dot')
#     name = 'tree' + str(i_tree)
#     graph.write_png(name+  '.png')
#     os.system('dot -Tpng tree.dot -o tree.png')
#     i_tree +=1
```

In [85]:

```
#importing the necessary libaries
import pandas as pd
import os
import six
import pydot
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import export_graphviz
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import seaborn as sns
```

In [17]:

```
wine_data = pd.read_csv("./Data/wine.data",header = None)
wine_data.columns  = ["Class Label","Alcohol","Malic acid","Ash","Alcalinity of
 ash","Magnesium","Total phenols","Flavanoids","Nonflavanoid phenols","Proanthoc
yanins","Color intensity","Hue","OD280/OD315 of diluted wines", "Proline"]
```

In [25]:

```
wine_data.head()
```

Out[25]:

	Class Label	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	P
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	

In [21]:

```
wine_data["Class Label"].nunique()
```

Out[21]:

```
3
```

In [22]:

```
wine_data["Class Label"].value_counts()
```

Out[22]:

```
2    71
1    59
3    48
Name: Class Label, dtype: int64
```

In [45]:

```
"""
    The dataset is relatively balanced, The number of data points per class are
 not so distant from each other.

"""

#getting the target variable
X = wine_data.drop('Class Label', axis=1)
y = wine_data['Class Label']

y.shape, X.shape
```

Out[45]:

```
((178,), (178, 13))
```

In [75]:

```
X
```

Out[75]:

	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proant
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	
...	...	...	...	...	...	...	...	...	
173	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	
174	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	
175	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	
176	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	
177	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	

178 rows × 13 columns

In [78]:

```
X
```

Out[78]:

	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proant
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	
...	...	...	...	...	...	...	...	...	
173	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	
174	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	
175	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	
176	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	
177	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	

178 rows × 13 columns

In [77]:

```
y.shape
```

Out[77]:

```
(178,)
```

In [70]:

```
#preparing the dataset for a machine learning algorithm
x_train, x_test, y_train, y_test = train_test_split(X, y, random_state = 7, test
_size = 0.25)
```

In [71]:

```
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

Out[71]:

```
((133, 13), (133,), (45, 13), (45,))
```

In [72]:

```
#scaling the data set
scaler = MinMaxScaler()
sc_x=scaler.fit(x_train)
x_train_scaled=sc_x.transform(x_train.values)
x_test_scaled=sc_x.transform(x_test.values)
```

In [73]:

```
x_train_scaled.shape
```

Out[73]:

```
(133, 13)
```

In [90]:

```
'''
We are going to run random forest on both the scaled data and the unscaled data
 to see which performs better
before we go onto tuning parameters
'''
rClassifier_1 = RandomForestClassifier()
rClassifier_2 = RandomForestClassifier()

rClassifier_1.fit(x_train,y_train)
y_preds = rClassifier_1.predict(x_test)

rClassifier_2.fit(x_train_scaled,y_train)
y_preds1 = rClassifier_2.predict(x_test_scaled)

print('The accuracy score for Random Forest Unscaled is', accuracy_score(y_test,
y_preds)*100,"%")
print('The accuracy score for Random Forest scaled is', accuracy_score(y_test,y_
preds1)*100,"%")
```

```
The accuracy score for Random Forest Unscaled is 100.0 %
The accuracy score for Random Forest scaled is 100.0 %
```

In [76]:

```
"""
    Random Forest Performs extremely well with the wine dataset with both unscal
ed and scaled data
"""
```

Out[76]:

```
'\n    Random Forest Performs extremely well with the wine dataset w
ith both unscaled and scaled data\n'
```

In [56]:

```
y_preds
```

Out[56]:

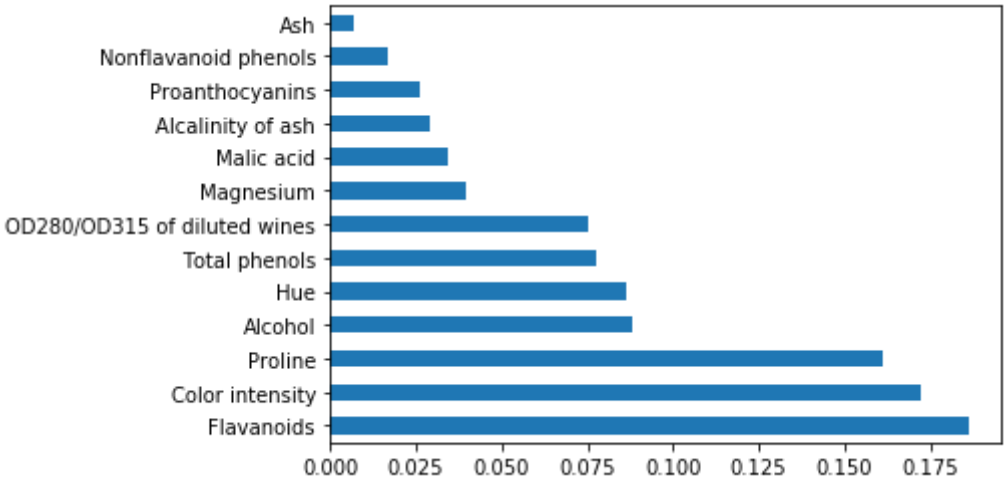
```
array([3, 1, 3, 3, 2, 3, 2, 1, 2, 3, 1, 2, 3, 2, 2, 2, 2, 3, 1, 1,
2, 2,
       2, 2, 1, 3, 2, 3, 3, 3, 2, 1, 3, 2, 2, 2, 3, 3, 1, 3, 1, 2,
3, 3,
       2])
```

In [80]:

```
#let us try to visualize the most important features
feat_importances = pd.Series(rClassifier_1.feature_importances_, index=X.columns
)
feat_importances.nlargest(13).plot(kind='barh')
```

Out[80]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f42ad2bd710>
```



In [ ]:

```
"""
According to the feature importance chart,
the most important criteria for predicting the classfier is Flavanoids.
"""
```

In [ ]:

```
estimator = model.estimators_[5]

from sklearn.tree import export_graphviz
# Export as dot file
export_graphviz(estimator, out_file='tree.dot',
                feature_names = iris.feature_names,
                class_names = iris.target_names,
                rounded = True, proportion = False,
                precision = 2, filled = True)

# Convert to png using system command (requires Graphviz)
from subprocess import call
call(['dot', '-Tpng', 'tree.dot', '-o', 'tree.png', '-Gdpi=600'])

# Display in jupyter notebook
from IPython.display import Image
Image(filename = 'tree.png')
```

In [86]:

```
X.columns
```

Out[86]:

```
Index(['Alcohol', 'Malic acid', 'Ash', 'Alcalinity of ash', 'Magnesi
um',
       'Total phenols', 'Flavanoids', 'Nonflavanoid phenols',
       'Proanthocyanins', 'Color intensity', 'Hue',
       'OD280/OD315 of diluted wines', 'Proline'],
      dtype='object')
```

In [88]:

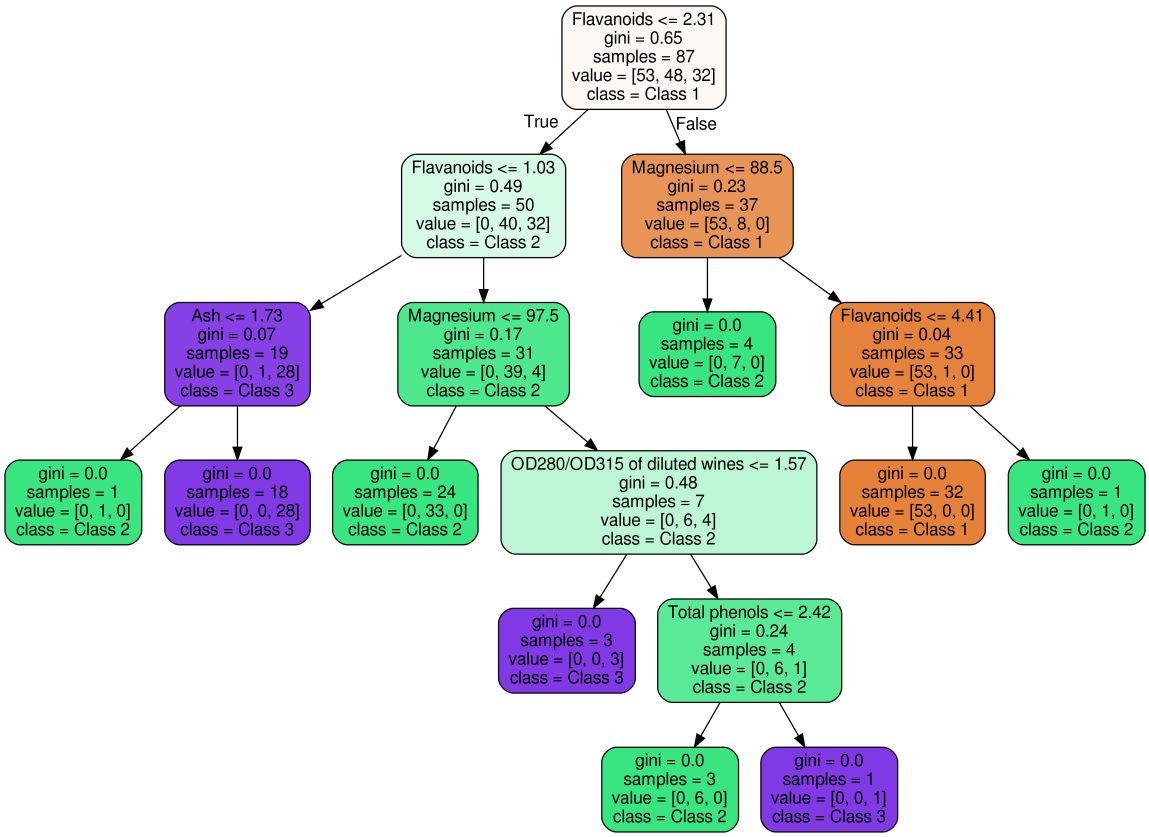
```
#Visualizing a single tree
estimator = rClassifier_1.estimators_[10]
target_names = ["Class 1", "Class 2", "Class 3"]

export_graphviz(estimator, out_file='tree.dot',
                feature_names = X.columns,
                class_names = target_names,
                rounded = True, proportion = False,
                precision = 2, filled = True)

# Convert to png using system command (requires Graphviz)
from subprocess import call
call(['dot', '-Tpng', 'tree.dot', '-o', 'tree.png', '-Gdpi=600'])

# Display in jupyter notebook
from IPython.display import Image
Image(filename = 'tree.png')
```

Out[88]:



In [ ]: