In [1]:
```
!pwd
!ls
```

```
/Users/xiasong/Documents/Class_2016/DSE/DSE220/homework/homework_1
Dataprocessing.ipynb  KNN.pdf             wine_train_data.csv
Dataprocessing.pdf    Untitled1.ipynb     wine_train_labels.csv
Decisiontree.ipynb    Untitled2.ipynb     wine_val_data.csv
Decisiontree.pdf      wine_modified.csv   wine_val_labels.csv
Homework_1.pdf        wine_test_data.csv
KNN.ipynb             wine_test_labels.csv
```

In [2]:
```
import csv
import pandas as pd
import numpy as np
import scipy as sp
from numpy import nan
```

The questions in this section are sequential steps. So use the data obtained after Question 1 for Question 2 and so on.

In [3]:
```
wmod = pd.read_csv('wine_modified.csv')
wmod.head()
```

Out[3]:

	class	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Pr
0	1.0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.2
1	1.0	13.20	1.78	NaN	11.2	100.0	2.65	2.76	0.26	1.2
2	1.0	13.16	2.36	NaN	18.6	101.0	2.80	3.24	0.30	2.8
3	1.0	14.37	NaN	2.50	NaN	NaN	3.85	NaN	NaN	Na
4	1.0	13.24	2.59	NaN	21.0	118.0	2.80	2.69	0.39	1.8

Question 1: Remove the rows with missing labels ('class') and rows with more than 7 missing features. Report the remaining number of rows. (1 mark)

Answer: the original number of rows is 178 and remaining number of rows is 154.

In [4]:
```
# create a function to find missing values
def num_missing(x):
    return sum(x.isnull())
len(wmod)
```

Out[4]: 178

In [5]:
```
wmodclrow=pd.DataFrame(columns=('class','Alcohol','Malic acid','Ash','Alcali
                                'Magnesium','Total phenols','Flavanoids','No
                                'Proanthocyanins','Color intensity','Hue','C
for i in range(len(wmod)):
    rows=wmod.iloc[i]
    if (pd.isnull(rows['class'])==False) and (num_missing(rows) <= 7):
        wmodclrow=wmodclrow.append(wmod.iloc[i])
```

In [122]:
```
wmodclrow.shape
```

Out[122]: (154, 14)

Question 2: Remove features with > 50% of missing values. For other fea- tures with missing values ll them with the mean of the corresponding features. Report the removed features (if any) and standard deviation of features with missing values after lling. (2 marks)

In [6]:
```
#look for the column with missing values larger than 50%
wmodclcol=wmodclrow
for column in wmodclcol:
    rownum = len(wmodclcol)
    if num_missing(wmodclcol[column]) > rownum/2:
        print (column)
```

```
Ash
```

In [7]:
```
#drop the column with missing values larger than 50%
wmodclcol=wmodclcol.drop(['Ash'], axis=1)
#fill other column's missing values with the mean of column
wmodclcolfil=wmodclcol.fillna(wmodclcol.mean())
```

In [8]:
```
print('The removed feature is "Ash" and the following is the standard deviat
      'features with missing value after filling')
wmodclcolfil.std()
```

```
The removed feature is "Ash" and the following is the standard deviation
 offeatures with missing value after filling
```

Out[8]:
```
class                     0.766522
Alcohol                   3.804067
Malic acid                1.116005
Alcalinity of ash         3.456794
Magnesium                14.440377
Total phenols             0.617237
Flavanoids                0.873573
Nonflavanoid phenols      0.127083
Proanthocyanins           0.587671
Color intensity           2.325204
Hue                       0.229412
OD280/OD315               0.723261
Proline                 303.033368
dtype: float64
```

Question 3: Detect and remove rows with any outliers/incorrect values in fea- tures 'alcohol' and 'proline' (if any). Clearly state the basis of your removal. (1 mark)

In [9]:
```
def replace(group):
    mean, std = group.mean(), group.std()
    outliers = (group - mean).abs() > 3*std
    group[outliers] = nan        # or "group[~outliers].mean()"
    return group
```

In [161]:
```
wmodclcolfil['Alcohol']=replace(wmodclcolfil['Alcohol'])
wmodclcolfil['Proline']=replace(wmodclcolfil['Proline'])
```

In [162]:
```
wmodclcolfilout=pd.DataFrame(columns=('class','Alcohol','Malic acid','Alcali
                                      'Total phenols','Flavanoids','Nonflava
                                      'Color intensity','Hue','OD280/OD315',
for i in range(len(wmodclcolfil)):
    rows=wmodclcolfil.iloc[i]
    if (pd.isnull(rows['Alcohol'])==False) and (pd.isnull(rows['Proline'])==
        wmodclcolfilout=wmodclcolfilout.append(wmodclcolfil.iloc[i])
```

In [166]:
```
print (len(wmodclcolfilout['Alcohol']),len(wmodclcolfil['Alcohol']))
```

```
148 154
```

In this problem, we remove the outliers based on the difference between value and feature mean larger than 3 times of feature's standard deviation

In [ ]:

In [2]:
```
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier, export_graphviz
import pydotplus
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
# Load the wine dataset
wtestdata = pd.read_csv('wine_test_data.csv')
wtestlabels = pd.read_csv('wine_test_labels.csv')
wtraindata = pd.read_csv('wine_train_data.csv')
wtrainlabels = pd.read_csv('wine_train_labels.csv')
wvaldata = pd.read_csv('wine_val_data.csv')
wvallabels = pd.read_csv('wine_val_labels.csv')
```

In [6]:
```
clf = DecisionTreeClassifier(criterion='gini')
clf.fit(wtraindata, wtrainlabels)
val_pred = clf.predict(wvaldata)
wvallab = list(wvallabels['class'])
print ('Validation accuracy = ' + str(np.sum(val_pred == wvallab)*1.0/len(va
```

```
Validation accuracy = 0.974358974359
```

In [5]:
```
clf = DecisionTreeClassifier(criterion='entropy')
clf.fit(wtraindata, wtrainlabels)
val_pred = clf.predict(wvaldata)
wvallab = list(wvallabels['class'])
print ('Validation accuracy = ' + str(np.sum(val_pred == wvallab)*1.0/len(va
```

```
Validation accuracy = 0.948717948718
```

According the accuracy of two criterions, we will use gini to train our model

In [7]:
```
#Using train data and validation data to training our model
X_trainframes = [wtraindata, wvaldata]
Y_trainframes = [wtrainlabels,wvallabels]
X_train = pd.concat(X_trainframes)
Y_train = pd.concat(Y_trainframes)
```

In [13]:
```
#The accuracy on the test data
clf = DecisionTreeClassifier(criterion='gini')
clf.fit(X_train, Y_train)
test_pred = clf.predict(wtestdata)
wtestlab = list(wtestlabels['class'])
print ('Test accuracy = ' + str(np.sum(test_pred == wtestlab)*1.0/len(test_p
```

```
Test accuracy = 0.769230769231
```

Our results showed that the accuracy on the validataion data is 95% by using Decision Tree model on train data for entropy criterions. However, the accuracy on the validation data is 97% when using gini criterions. According to our results we adopted entropy criteerions to predict the test data.

When we use trained model to predict test data, the accuracy attained 77%.

Question 5: Use the criterion selected above to train Decision Tree model on train data for min samples split=f2,5,10,20g and report the accuracies on the validation data. Select the best parameter and report the accuracy on the test data. (2 marks)

In [16]:
```
#min_sample_split=2,5,10,20
for i in (2,5,10,20):
    clf = DecisionTreeClassifier(criterion='gini', min_samples_split=i)
    clf.fit(wtraindata, wtrainlabels)
    val_pred = clf.predict(wvaldata)
    wvallab = list(wvallabels['class'])
    print ('when min_sample_split is %s' % i)
    print ('we get the Validation accuracy = ' + str(np.sum(val_pred == wval
```

```
when min_sample_split is 2
we get the Validation accuracy = 0.897435897436
when min_sample_split is 5
we get the Validation accuracy = 0.974358974359
when min_sample_split is 10
we get the Validation accuracy = 0.948717948718
when min_sample_split is 20
we get the Validation accuracy = 0.923076923077
```

In [17]:
```
#The accuracy on the test data
clf = DecisionTreeClassifier(criterion='gini', min_samples_split=5)
clf.fit(X_train, Y_train)
test_pred = clf.predict(wtestdata)
wtestlab = list(wtestlabels['class'])
print ('Test accuracy = ' + str(np.sum(test_pred == wtestlab)*1.0/len(test_p
```

```
Test accuracy = 0.74358974359
```

When we use min_sample_split in 2,5,10, and 20,the accuracy on the validataion data is 90%, 97%, 95% and 92% respectively. Therefore, we select min_sample_split 5 as our parameters in our model. When we use trained model to predict test data, the accuracy attained 74%.

Question 6: Use the parameters selected above (Q4 and Q5) to train Decision Tree model using the rst 20, 40, 60, 80 and 100 samples from train data. Keep the validation set unchanged during this analysis. Report and plot the accuracies on the validation data. (2 marks)

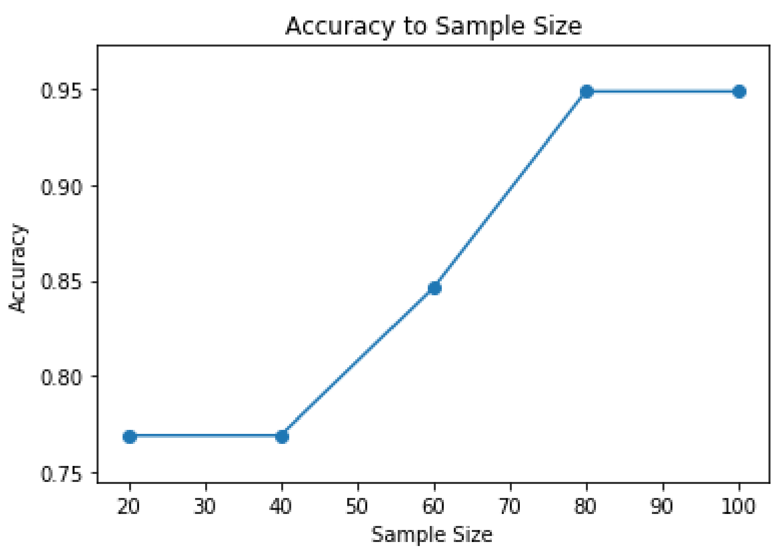
In [18]:

```
#train data using the first 20 samples
accu=pd.DataFrame(columns=['Sample','accuracy'])
for i in (20,40,60,80,100):
    data = wtraindata.head(i)
    labels = wtrainlabels.head(i)
    clf = DecisionTreeClassifier(criterion='gini', min_samples_split=5)
    clf.fit(data, labels)
    val_pred = clf.predict(wvaldata)
    accuracy = np.sum(val_pred == wvallab)*1.0/len(val_pred)
    data = pd.DataFrame({'Sample': [i],'accuracy':[accuracy]})
    accu=accu.append(data)
    wvallab = list(wvallabels['class'])
    print ('when our samples are first %s' % i)
    print ('We get validation accuracy = ' + str(np.sum(val_pred == wvallab)
```

```
when our samples are first 20
We get validation accuracy = 0.769230769231
when our samples are first 40
We get validation accuracy = 0.769230769231
when our samples are first 60
We get validation accuracy = 0.846153846154
when our samples are first 80
We get validation accuracy = 0.948717948718
when our samples are first 100
We get validation accuracy = 0.948717948718
```

In [19]:

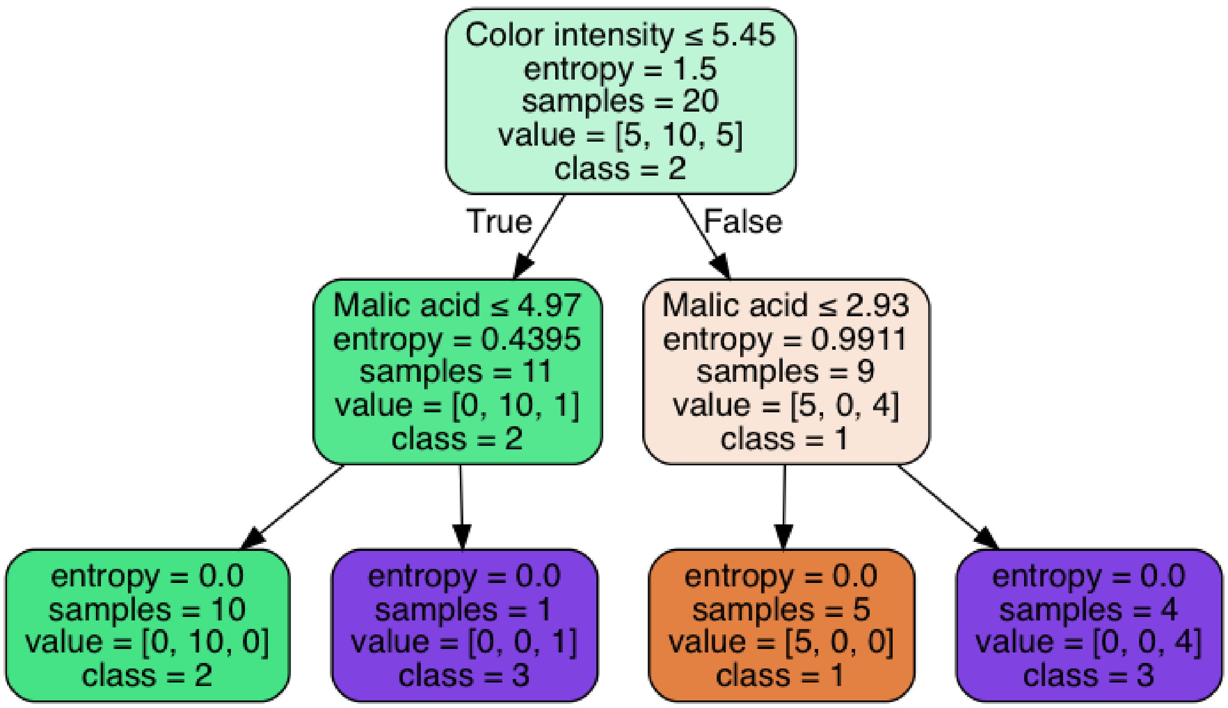
```
#plot accuracy with k value
plt.scatter(accu['Sample'], accu['accuracy'])
plt.plot(accu['Sample'], accu['accuracy'])
plt.xlabel('Sample Size')
plt.ylabel('Accuracy')
plt.title("Accuracy to Sample Size")
plt.show()
```



In [4]:
```
from IPython.display import Image
#train data using the first 20 samples
data = wtraindata.head(20)
labels = wtrainlabels.head(20)
clf = DecisionTreeClassifier(criterion='entropy', min_samples_split=5)
clf.fit(data, labels)
dot_data = export_graphviz(clf, out_file=None,
                          feature_names=wtraindata.columns,
                          class_names=['1','2','3'],
                          filled=True, rounded=True,
                          special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data)
print('first 20 samples')
Image(graph.create_png())
```

```
first 20 samples
```

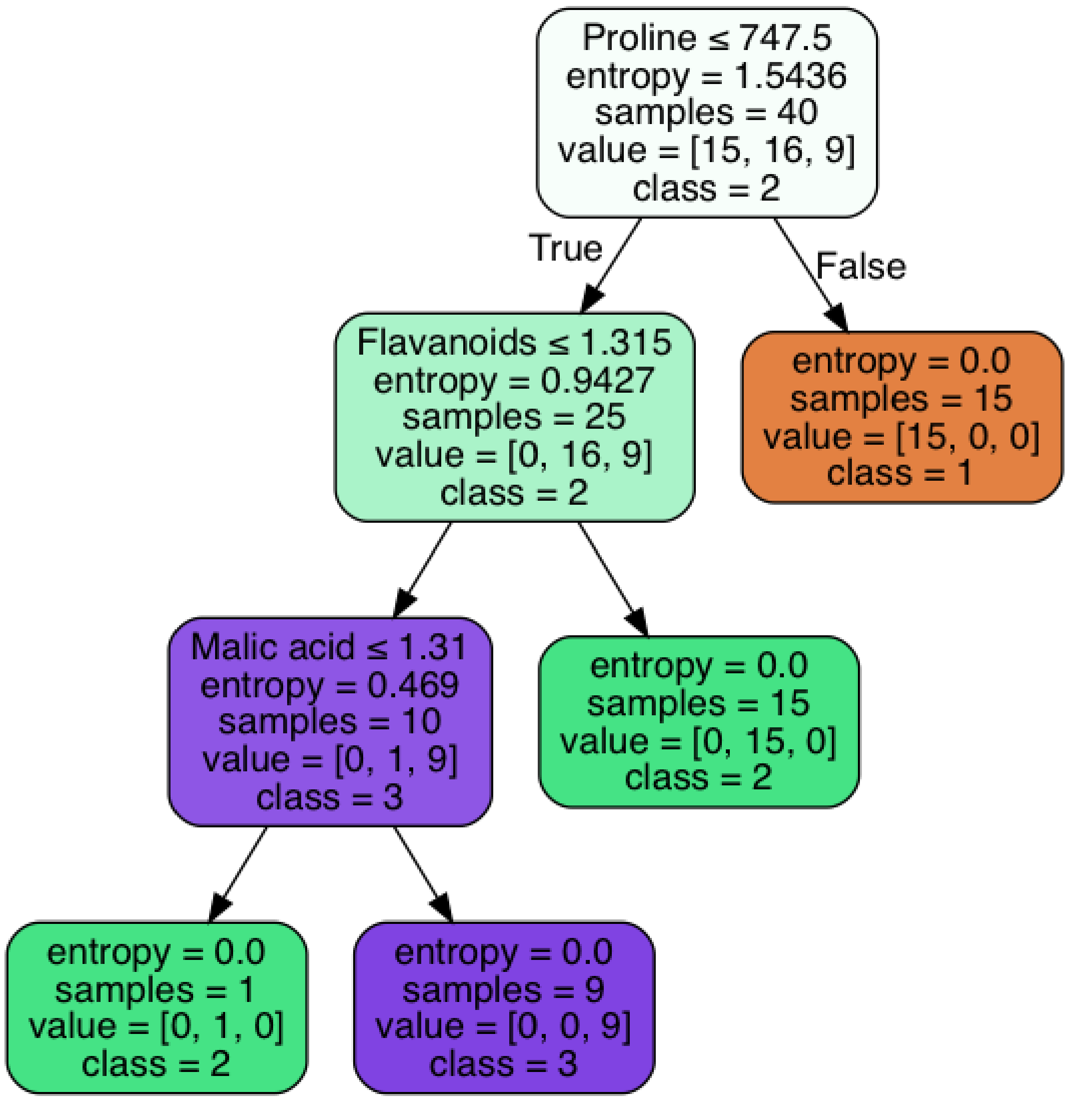
Out[4]:



In [5]:
```
#train data using the first 40 samples
data = wtraindata.head(40)
labels = wtrainlabels.head(40)
clf = DecisionTreeClassifier(criterion='entropy', min_samples_split=5)
clf.fit(data, labels)
dot_data = export_graphviz(clf, out_file=None,
                           feature_names=wtraindata.columns,
                           class_names=['1','2','3'],
                           filled=True, rounded=True,
                           special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data)
print('first 40 samples')
Image(graph.create_png())
```

```
first 40 samples
```

Out[5]:

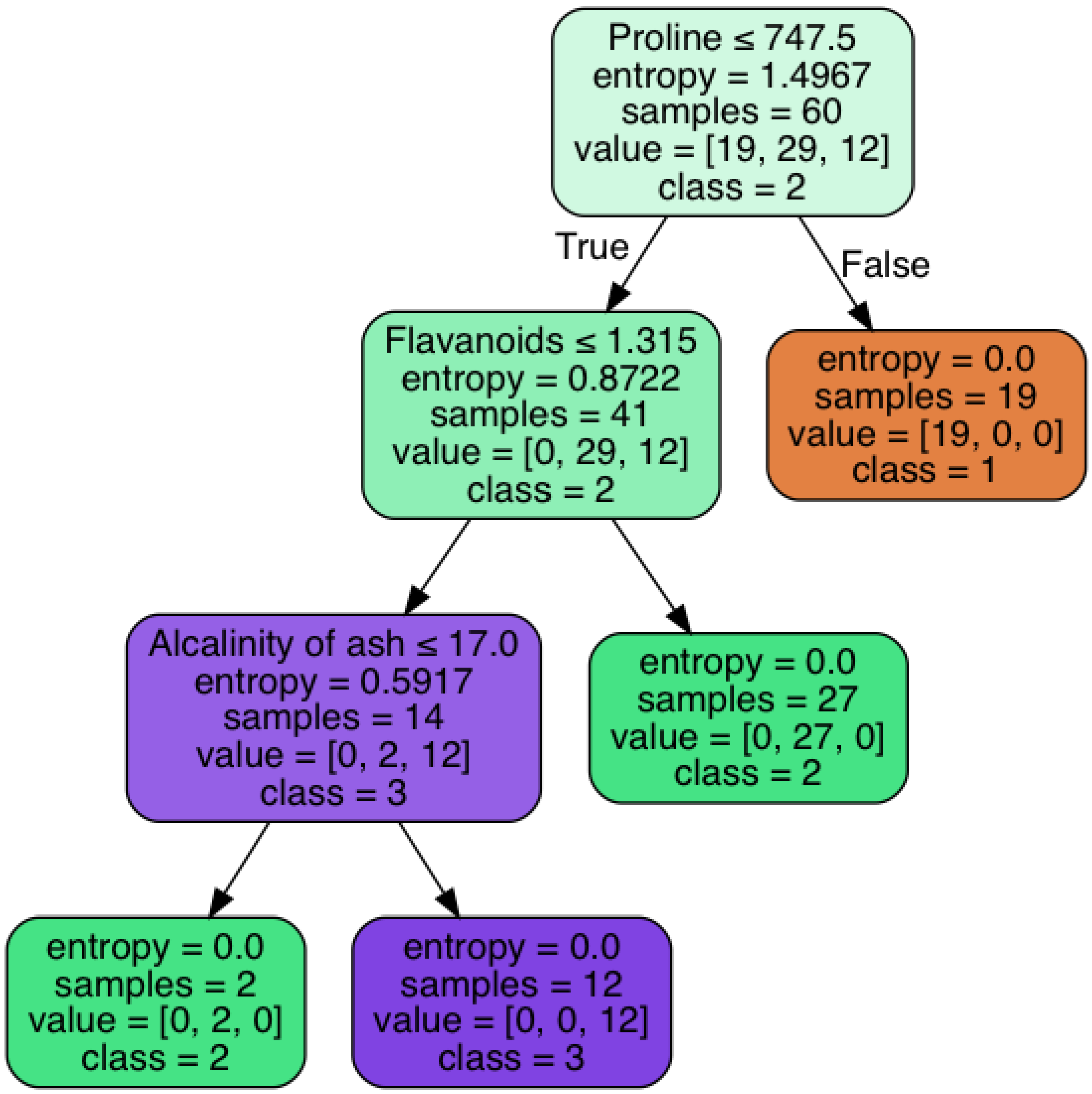


In [6]:

```
#train data using the first 60 samples
data = wtraindata.head(60)
labels = wtrainlabels.head(60)
clf = DecisionTreeClassifier(criterion='entropy', min_samples_split=5)
clf.fit(data, labels)
dot_data = export_graphviz(clf, out_file=None,
                           feature_names=wtraindata.columns,
                           class_names=['1','2','3'],
                           filled=True, rounded=True,
                           special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data)
print('first 60 samples')
Image(graph.create_png())
```

```
first 60 samples
```

Out[6]:

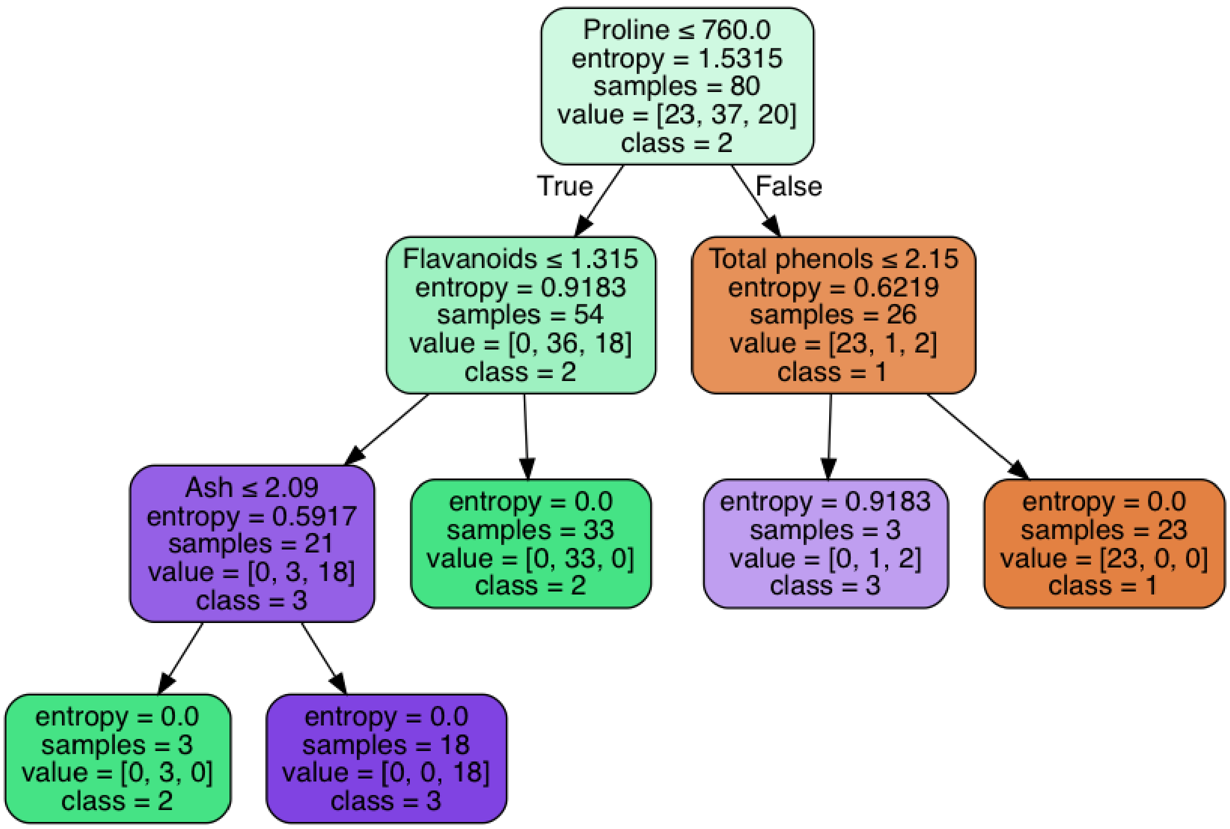


In [7]:

```
#train data using the first 80 samples
data = wtraindata.head(80)
labels = wtrainlabels.head(80)
clf = DecisionTreeClassifier(criterion='entropy', min_samples_split=5)
clf.fit(data, labels)
dot_data = export_graphviz(clf, out_file=None,
                           feature_names=wtraindata.columns,
                           class_names=['1','2','3'],
                           filled=True, rounded=True,
                           special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data)
print('first 80 samples')
Image(graph.create_png())
```

```
first 80 samples
```

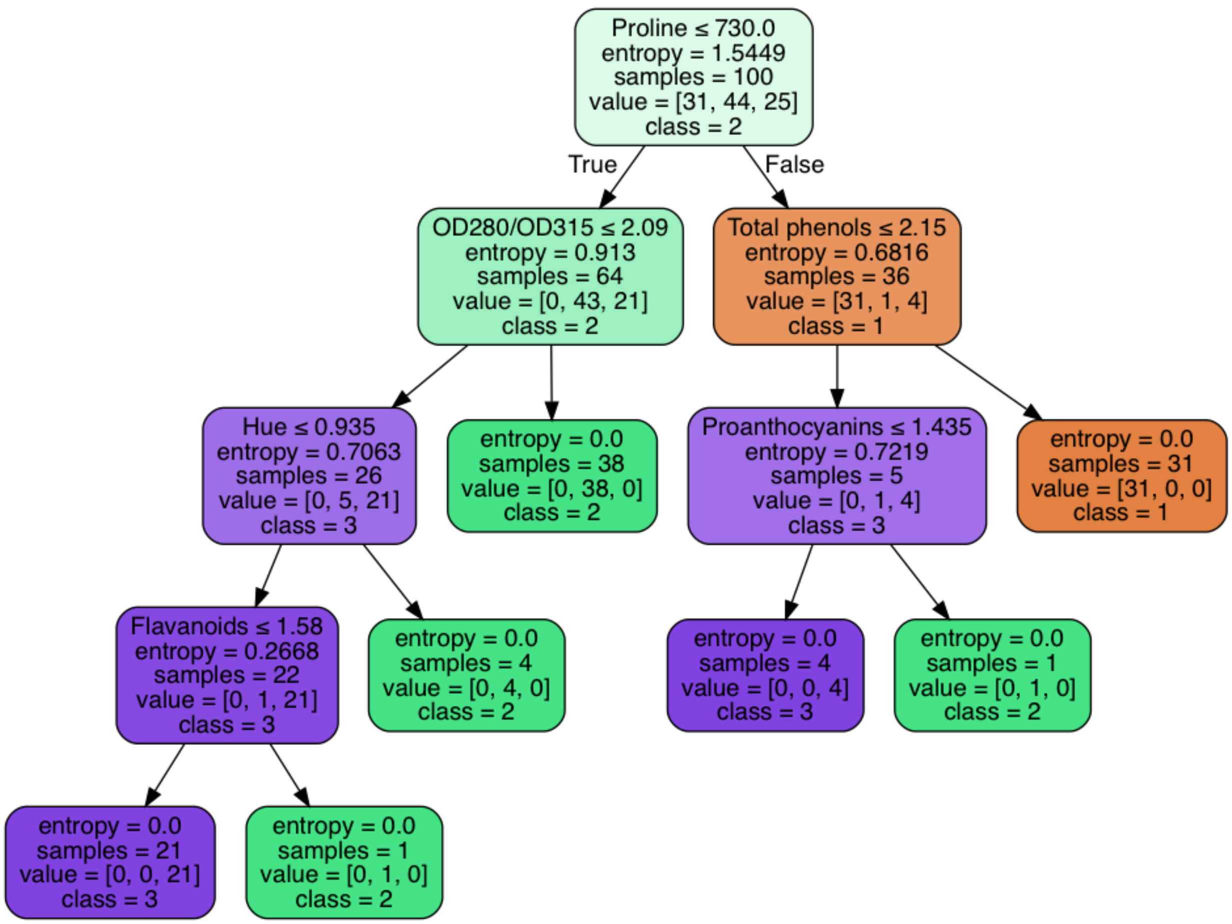
Out[7]:



In [8]:
```
#train data using the first 100 samples
data = wtraindata.head(100)
labels = wtrainlabels.head(100)
clf = DecisionTreeClassifier(criterion='entropy', min_samples_split=5)
clf.fit(data, labels)
dot_data = export_graphviz(clf, out_file=None,
                           feature_names=wtraindata.columns,
                           class_names=['1','2','3'],
                           filled=True, rounded=True,
                           special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data)
print('first 100 samples')
Image(graph.create_png())
```

```
first 100 samples
```

Out[8]:



In [ ]:

In [65]:
```
for i in ('manhattan','chebyshev','euclidean'):
    clf = KNeighborsClassifier(3, metric=i)
    clf.fit(wtraindata_norm, wtrainlabels.values.ravel())
    predictions = clf.predict(wvaldata_norm)
    wvallab = list(wvallabels['class'])
    accuracy = np.sum(predictions == wvallab)/(len(predictions))
    print ("Accuracy = " + str(accuracy) + " using distance metric %s" %i)
```

```
Accuracy = 0.948717948718 using distance metric manhattan
Accuracy = 0.923076923077 using distance metric chebyshev
Accuracy = 0.923076923077 using distance metric euclidean
```

Results showed us that the manhattan distance is te best metric for our validation data

In [52]:
```
#test data predition
clf = KNeighborsClassifier(3, metric='manhattan')
clf.fit(wtraindata_norm, wtrainlabels.values.ravel())
predictions = clf.predict(wtestdata_norm)
wtestlab = list(wtestlabels['class'])
accuracy = np.sum(predictions == wtestlab)/(len(predictions))
print ("Accuracy = " + str(accuracy) + " at k = 3")
```

```
Accuracy = 0.948717948718 at k = 3
```

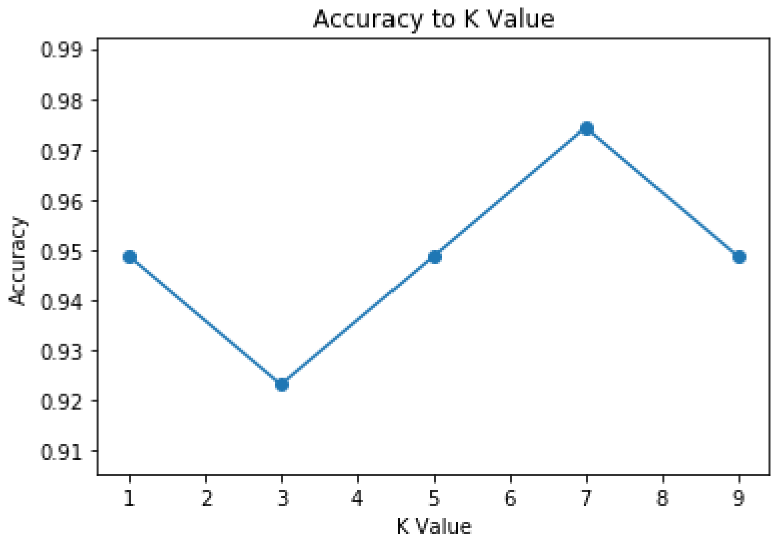
Question 9: Train the k-nn model on train data for k=1,3,5,7,9. Report and plot the accuracies on the validation data. Select the best 'k' value and report the accuracy on the test data for the selected 'k'. Use Euclidean distance. (2 marks)

In [84]:
```
accu=pd.DataFrame(columns=['kvalue','accuracy'])
for i in (1,3,5,7,9):
    clf = KNeighborsClassifier(i, p=2)
    clf.fit(wtraindata_norm, wtrainlabels.values.ravel())
    predictions = clf.predict(wvaldata_norm)
    wvallab = list(wvallabels['class'])
    accuracy = np.sum(predictions == wvallab)/(len(predictions))
    data = pd.DataFrame({'kvalue': [i],'accuracy':[accuracy]})
    accu=accu.append(data)
    print ("Accuracy = " + str(accuracy) + " at k = %s" % i)
```

```
Accuracy = 0.948717948718 at k = 1
Accuracy = 0.923076923077 at k = 3
Accuracy = 0.948717948718 at k = 5
Accuracy = 0.974358974359 at k = 7
Accuracy = 0.948717948718 at k = 9
```

Results showed us when we use k = 7, we got the highest accuracy for our validation data

In [90]:
```
#plot accuracy with k value
plt.scatter(accu['kvalue'], accu['accuracy'])
plt.plot(accu['kvalue'], accu['accuracy'])
plt.xlabel('K Value')
plt.ylabel('Accuracy')
plt.title("Accuracy to K Value")
plt.show()
```



In [55]:
```
clf = KNeighborsClassifier(7, p=2)
clf.fit(wtraindata_norm, wtrainlabels.values.ravel())
predictions = clf.predict(wtestdata_norm)
wtestlab = list(wtestlabels['class'])
accuracy = np.sum(predictions == wtestlab)/(len(predictions))
print ("Accuracy = " + str(accuracy) + " at k=7")
```

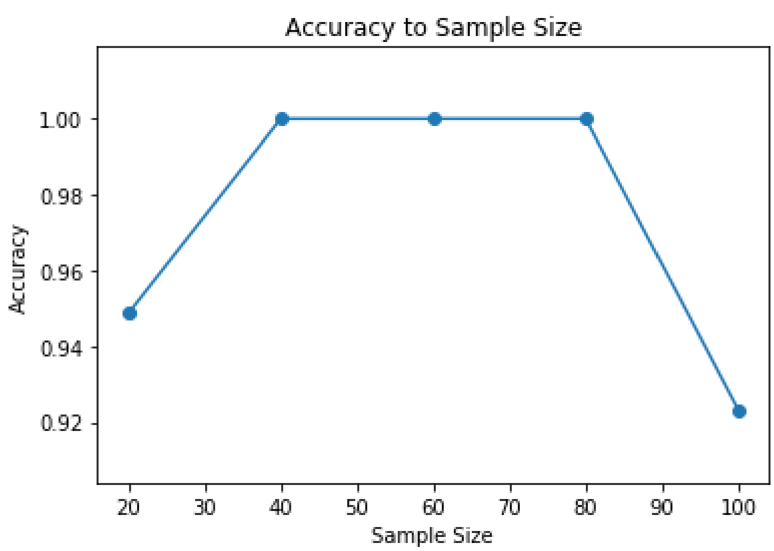
```
Accuracy = 0.948717948718 at k=7
```

Question 10: Instead of using full train data, train the model using the rst 20, 40, 60, 80 and 100 data samples from train data. Keep the validation set unchanged during this analysis. Report and plot the accuracies on the validation data. Use Euclidean distance and k=3. Note: Don't shue the data and use only the 'rst n samples', otherwise your answers may di er. (2 marks)

In [97]:
```
accu=pd.DataFrame(columns=['Sample','accuracy'])
for i in (20,40,60,80,100):
    data = wtraindata_norm.head(i)
    labels = wtrainlabels.head(i)
    clf = KNeighborsClassifier(3, p=2)
    clf.fit(data, labels.values.ravel())
    predictions = clf.predict(wvaldata_norm)
    wvallab = list(wvallabels['class'])
    accuracy = np.sum(predictions == wvallab)/(len(predictions))
    data = pd.DataFrame({'Sample': [i],'accuracy':[accuracy]})
    accu=accu.append(data)
    print ('when our samples are %s:' % i)
    print ('We get validation accuracy = ' + str(accuracy))
```

```
when our samples are 20:
We get validation accuracy = 0.948717948718
when our samples are 40:
We get validation accuracy = 1.0
when our samples are 60:
We get validation accuracy = 1.0
when our samples are 80:
We get validation accuracy = 1.0
when our samples are 100:
We get validation accuracy = 0.923076923077
```

In [95]:
```
#plot accuracy with sample size
plt.scatter(accu['Sample'], accu['accuracy'])
plt.plot(accu['Sample'], accu['accuracy'])
plt.xlabel('Sample Size')
plt.ylabel('Accuracy')
plt.title("Accuracy to Sample Size")
plt.show()
```



In [ ]: