# Exercises

## Sol 2:

**2.1** Gini index=1 − Σ i=0c − 1pi(t)^2,

pi(t) = 0.5

Gini = 1- 2*(0.5)^2 = **0.5**

**2.2 Gini index for the Customer ID attribute**

Gini for 1$^{st}$ customer ID 1 is Gini = 1-(1/1)^2-(0/1)^2 = 0

So on Gini will be zero for all, that means, Gini index for the Customer ID attribute is **0 Zero**

**2.3 Compute the Gini index for the Gender attribute.**

For Gender we should divide first Male and Female:

We have 50% ratio of Male and Female.

For Male Gini_M = 1-2*(0.5)^2 = 0.5

For female Gini_F = 1-2*(0.5)^2 = 0.5

Gini_total = 0.5*Gini_M + 0.5*Gini_F = 0.5*0.5 + 0.5*0.5 = **0.5**

**2.4 Compute the Gini index for the Car Type attribute using multiway split.**

we have 3 splits: Family, Sport, and Luxury.

For Family we have 3 C1 and 1 C0 then our Gini will be: 1 – (1/4)^2 – (3/4)^2 = **0.375**

For Sport we have all C0 so our Gini is **0**

For Luxury we have 1 C0 and 7 C1 our Gini will be 1-(1/8)^2 – (7/8)^2 = **0.2188**

Overall Gini for Car Type = 4/20*0.375 + 0 + 8/20*0.2188 = **0.1625**

**2.5 Compute the Gini index for the Shirt Size attribute using multiway split.**

Small shirt size we have 3 C0 and 2 C1 our Gini will be **0.48**

Medium shirt size we have 3 C0 and 4 C1 our Gini will be **0.4898**

Large shirt size we have 2 C0 and 2 C1 our Gini will be **0.5**

Extra Large shirt we have 2 C0 and 2 C1 our Gini will be **0.5**

Overall gini for Shirt Size attribute is **0.4914**

**2.6 Which attribute is better, Gender, Car Type, or Shirt Size?**

Lower value will be the better one, for **Car Type** we have min Gini: 0.1625

**2.7 Explain why Customer ID should not be used as the attribute test condition even though it has the lowest Gini.**

Because Customer ID is not meaning full in terms of model. ID is just auto increasing function.

## Sol 3:

$$Entropy(t) = -\sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t)$$

**3.1 What is the entropy of this collection of training examples with respect to the class attribute?**

+ target class = 4 then p(+) = 4/9

– target class = 5 then p(-) = 5/9

Entropy= −4/9 log2(4/9) − 5/9 log2(5/9) = **0.9911**

**3.2 What are the information gains of a1 and a2 relative to these training examples?**

Let first find the entropy for each then will subtract from the overall entropy.

Information Gain(a1) = Entropy Total – Entropy(a1)

Entropy(a1): In a1 we have 2 split T and F. In T we have 3+ and 1- and in F we have 1+ and 4-

4/9 [− (3/4) log2(3/4) − (1/4) log2(1/4)] + 5/9 [ − (1/5) log2(1/5) − (4/5) log2(4/5)] = **0.7616**

**Information Gain: 0.9911 – 0.7616 = 0.2294**

Similarly, for a2 **Information Gain: 0.9911 – 0.9839 = 0.0072**

**3.3 For a3, which is a continuous attribute, compute the information gain for every possible split.**

| a3 | Class label | Split Point | Entropy | Information Gain |
|---|---|---|---|---|
| 1.0 | + | 2.0 | 0.8484 | 0.1427 |
| 3.0 | - | 3.5 | 0.9885 | 0.0026 |
| 4.0 | + | 4.5 | 0.9183 | 0.0728 |
| 5.0 | - | | | |
| 5.0 | - | 5.5 | 0.9839 | 0.0072 |
| 6.0 | + | 6.5 | 0.9728 | 0.0183 |
| 7.0 | + | | | |
| 7.0 | - | 7.5 | 0.8889 | 0.1022 |

We can see we have highest information gain at split point **2.0 for a3**

**3.4 What is the best split (among a1, a2, and a3) according to the information gain?**

From the previous 2 question we found that we have highest information gain for a1.

So we can say a1 is the best split for this data set.

**3.5 What is the best split (between a1 and a2) according to the misclassification error rate?**

Misclassification rate = 1 – max(pi)

For a1 we have error rate = 2/9
For a2 we have error rate = 4/9

According to error rate **a1 is the best split** because we have less error rate for a1.

**3.6 What is the best split (between a1 and a2) according to the Gini index?**

For a1, the Gini: $4/9 *(1 - (3/4)^2 - (1/4)^2) + 5/9*(1 - (1/5)^2 - (4/5)^2) =$ **0.3444**.

For a2, the Gini: $5/9 *(1 - (2/5)^2 - (3/5)^2) + 4/9*(1 - (2/4)^2 - (2/4)^2) =$ **0.4889**.

Now we can say Gini index for a1 is lesser then a2 that means **a1 split is the best**.

## Sol 5:

**5.1 Calculate the information gain when splitting on A and B. Which attribute would the decision tree induction algorithm choose?**

For A

| A | T | F |
|---|---|---|
| + | 4 | 0 |
| - | 3 | 3 |

For B

| B | T | F |
|---|---|---|
| + | 3 | 1 |
| - | 1 | 5 |

Entropy: -4/10*log2(4/10) -6/10*log2(6/10) = **0.9710**

Information gain for A after split:

Entropy_A_T = -4/7*log(4/7) -3/7*log(3/7) = 0.9852

Entropy_A_F = -3/3*log(3/3) = 0

Information gain for A = E – 7/10* Entropy_A_T – 3/10* Entropy_A_F

= 0.9710 – 7/10*(0.9852) = **0.2813**

Similarly,

E_B_T = 0.8113

E_B_F = 0.6500

Information gain for B = E – 4/10*E_B_T – 6/10*E_B_F = **0.2565**

Here we can say that **A has higher Information gain and chosen.**

**5.2Calculate the gain in the Gini index when splitting on *A* and *B*. Which attribute would the decision tree induction algorithm choose?**

Over all Gini: G_o = 1- (4/10)^2 – (6/10)^2 = 0.48

Info gain after split on A:

G_A_T= 1-(4/7)^2-(3/7)^2 = 0.489

G_A_F = 1-(3/3)^2 – (0/3)^2 = 0

Gain_A = G_o – 7/10*G_A_T – 0 = **0.137**

Similarly, Gain_B = **0.1633**

**5.3 shows that entropy and the Gini index are both monotonically increasing on the range [0, 0.5] and they are both monotonically decreasing on the range [0.5, 1]. Is it possible that information gain and the gain in the Gini index favor different attributes? Explain.**

From part 5a: Gain_A = 0.2813, Gain_B = 0.2565

From part 5b: Gain_A = 0.137, Gain_B = 0.1633

Yes, though these measures have similar range and monotonous behaviors, their gains do not necessarily behave **in** the same way.

## Sol 6:

| Class | P | C1 | C2 |
|---|---|---|---|
| Class 0 | 7 | 3 | 4 |
| Class 1 | 3 | 0 | 3 |

1. Gini at parent node: 1-(7/10)^2 – (3/10)^2 = **0.42**
   Error Rate: 1-max(7/10,3/10) = 1-7/10 = **0.3**

2. Gini Index at Child node:
   Gini(C1) = 1 – (3/3)^2-(0/3)^2 = 0
   Gini(C2) = 1-(4/7)^2-(3/7)^2 = 0.489
   Gini(Children) = 3/10*0+7/10*0.489 = 0.342

   Yes, I would consider Gini for that optimizes certain criterion in team of impurity measure.

3. Error rate at child nodes:
   Error(c1) = 1-max(3/3,0) = 1-1 = **0**
   Error(c2) = 1-max(4/7,3/7) = 1-4/7 = **0.42857**

   Yes, I would consider Misclassification rate for that optimizes certain criterion in team of impurity measure.

Homework 2

Solution 7a.

a) For Level 1:

For X,

| X | C1 | C2 |
|---|---|---|
| 0 | 60 | 60 |
| 1 | 40 | 40 |

Error Rate(X) = (60+40)/200 =0.5

For Y,

| Y | C1 | C2 |
|---|---|---|
| 0 | 40 | 60 |
| 1 | 60 | 40 |

Error Rate(Y) = (40+40)/200 =0.4

For Z,

| Z | C1 | C2 |
|---|---|---|
| 0 | 30 | 70 |
| 1 | 70 | 30 |

Error Rate(Z) = (30+30)/200 =0.3

Z is chosen

b) For Level 2: we have two factor, Z=0 and Z=1

For Z=0,

| X | C1 | C2 |
|---|---|---|
| 0 | 15 | 45 |
| 1 | 15 | 25 |

| Y | C1 | C2 |
|---|---|---|
| 0 | 15 | 45 |
| 1 | 15 | 25 |

Error Rate(X)+(Y) = (15+15)/100 =0.3

For Z=1,

| X | C1 | C2 |
|---|---|---|
| 0 | 45 | 15 |
| 1 | 25 | 15 |

| Y | C1 | C2 |
|---|---|---|
| 0 | 25 | 15 |
| 1 | 45 | 15 |

Error Rate(X)+(Y) = (15+15)/100 =0.3

Overall Error rate = (15+15+15+15)/200 =0.3

Solution 7b. X is the splitting attribute, whereas Y and Z be the test condition attribute

For X=0,

| Y | C1 | C2 |
|---|---|---|
| 0 | 5 | 55 |
| 1 | 55 | 5 |

| Z | C1 | C2 |
|---|---|---|
| 0 | 15 | 45 |
| 1 | 45 | 15 |

Error Rate (Y) = (5+5)/120 = 1/12
Error Rate (Z) = (15+15)/120 = 1/4 -> better split

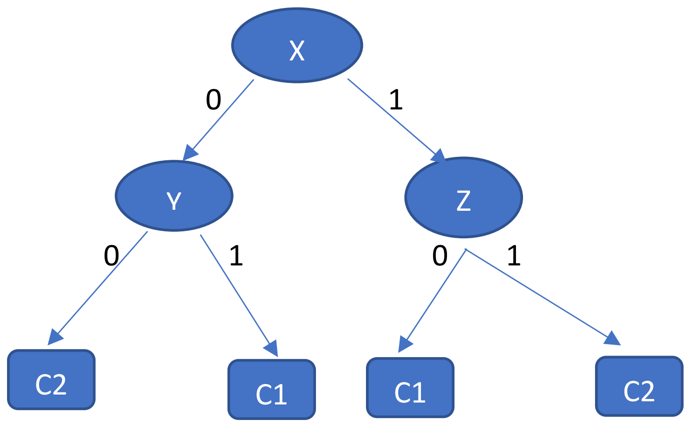
For X=1,

| Y | C1 | C2 |
|---|---|---|
| 0 | 35 | 5 |
| 1 | 5 | 35 |

| Z | C1 | C2 |
|---|---|---|
| 0 | 15 | 25 |
| 1 | 25 | 15 |

Error Rate (Y) = (5+5)/80 = 1/8. -> better split

Error Rate (Z) = (15+15)/80 = 3/8




Overall Error Rate = (10+10)/200 =0.1

Solution 7c. We can see the error rate in part a is slightly higher than part b. This implies that greedy heuristic does not always give optimal solution.

Solution 8a.

Overall Error Rate (before split)

$E_0$ = 1-max(50/100,50/100) = 50/100

After splitting on A:

| A | T | F |
|---|---|---|
| + | 25 | 25 |
| - | 0 | 50 |

Error Rate $E_{AT}$ = 1-max(25/25,0/25) = 0/25 = 0

Error Rate $E_{AF}$ = 1-max(25/75,50/75) = 25/75

$\Delta_A = E_0$ - 25/100 $E_{AT}$ -75/100 $E_{AF}$ = 50/100-0-75/100*25/75 = 25/100

After splitting on B:

| B | T | F |
|---|---|---|
| + | 30 | 20 |
| - | 20 | 30 |

Error Rate $E_{BT}$ = 1-max(30/50,20/50) = 20/50

Error Rate $E_{BF}$ = 1-max(20/50,30/50) = 20/50

$\Delta_B = E_0$ - 25/100 $E_{BT}$ -75/100 $E_{BF}$ = 50/100-50/100*20/50-50/100*20/50 = 10/100

After splitting on C:

| C | T | F |
|---|---|---|
| + | 25 | 25 |
| - | 25 | 25 |

Error Rate $E_{CT}$ = 25/50

Error Rate $E_{CF}$ = 25/50

$\Delta_{C} = E_{0}$ - 25/100 $E_{CT}$ -75/100 $E_{CF}$ = 50/100-50/100*25/50-50/100*25/50 = 0

Hence, we choose ‘A’ because we have highest gain.

Solution 8b. Repeat for two children of the root node.

$A_T$ child node is pure -> no splitting needed.

For $A_F$,

| B | C | + | - |
|---|---|---|---|
| T | T | 0 | 20 |
| F | T | 0 | 5 |
| T | F | 25 | 0 |
| F | F | 0 | 25 |

Classification Error $A_F = E_0$ = 25/75

After splitting on B:

| B | T | F |
|---|---|---|
| + | 25 | 0 |
| - | 20 | 30 |

Error Rate $E_{BT}$ = 20/45

Error Rate $E_{BF}$ = 0

$\Delta_{B} = E_{0}$ - 45/75 $E_{BT}$ -30/75 $E_{BF}$ = 25/75 - 45/75*20/45 -30/75*0 = 5/75

After splitting on C:

| C | T | F |
|---|---|---|
| + | 0 | 25 |
| - | 25 | 25 |

Error Rate $E_{CT}$ = 0/25

Error Rate $E_{CF}$ = 25/50

$\Delta_{C} = E_{0}$ - 25/75 $E_{CT}$ -50/75 $E_{CF}$ = 25/75-25/75*0/25-50/75*25/50 = 0

Hence, gain in C is zero that means will split on B.

Solution 8c. 20 instances has been misclassified by the resulting decision tree.

Solution 8d. Repeat part a,b and c. Using c as the splitting attribute.

For $C_T$,

$E_0 = 25/50$

After splitting on A:

| A | T | F |
|---|---|---|
| + | 25 | 0 |
| - | 0 | 25 |

Error Rate $E_{AT} = 0/25 = 0$

Error Rate $E_{AF} = 0/25 = 0$

$\Delta_A = 25/50$

After splitting on B:

| B | T | F |
|---|---|---|
| + | 5 | 20 |
| - | 20 | 5 |

Error Rate $E_{BT} = 5/25$

Error Rate $E_{BF} = 5/25$

$\Delta_B = 15/50$

Hence, the gain of A is higher, so we choose A.

For $C_F$,

$E_0 = 25/50$

After splitting on A:

| A | T | F |
|---|---|---|
| + | 0 | 25 |
| - | 0 | 25 |

Error Rate $E_{AT} = 0$

Error Rate $E_{AF} = 25/50$

$\Delta_A = 0$

After splitting on B:

| B | T | F |
|---|---|---|
| + | 25 | 0 |
| - | 0 | 25 |

Error Rate $E_{BT} = 0/25 = 0$

Error Rate $E_{BF} = 0$
$\Delta_B = 25/50$

So B is the splitting attribute.

Solution 8e. The greedy heuristic method does not lead to the best tree.

## Sol 12:

Table 3.7

| | Accuracy | |
|---|---|---|
| Data Set | T10 | T100 |
| A | 0.86 | 0.97 |
| B | 0.84 | 0.77 |

12.1 Based on the accuracies shown in Table 3.7, which classification model would you expect to have better performance on unseen instances?

- Since we have data set B is unseen for model which trained on Data Set A. We can see in data set B(Unseen) is 0.84 in **model T10** which is better then T100. Model T10 is better performance on unseen instances.

12.2 Now, you tested T10 and T100 on the entire data set (A+B) and found that the classification accuracy of T10 on data set (A+B) is 0.85, whereas the classification accuracy of T100 on the data set (A+B) is 0.87. Based on this new information and your observations from Table 3.7, which classification model would you finally choose for classification?

| Data Set | T10 | T100 |
|---|---|---|
| A+B | 0.85 | 0.87 |

- In this case we have T10 with accuracy 0.85 and T100 with 0.87. Then I would like to go with **T10** because we do not have much difference in accuracy just 0.02% but our model will be simpler and less overfitted as compare to big and complex model which is T100 with 100 leaf nodes and high chance of overfitting that we already noticed in data set B model T100 accuracy 0.77.

### Problem 2.1

In [76]:
```
import numpy as np
import pandas as pd

from scipy import stats

from sklearn.datasets import load_iris

from sklearn import tree

from sklearn import model_selection
from sklearn import metrics

import matplotlib.pyplot as plt

%matplotlib inline

import graphviz
```

In [77]:
```
iris = load_iris()
print(iris.keys())
df=pd.DataFrame(iris.data, columns=iris.feature_names)
y=iris.target

X_train, X_test, Y_train, Y_test = model_selection.train_test_split(df, y, tes
t_size=0.2, random_state = 0)
```

```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filen
ame'])
```

For Depth = 1

In [78]:
```
distree1 = tree.DecisionTreeClassifier(criterion='gini', max_depth=1, min_samp
les_leaf=2)
distree1.fit(X_train, Y_train)
predicted_distree1 = distree1.predict(X_test)
```

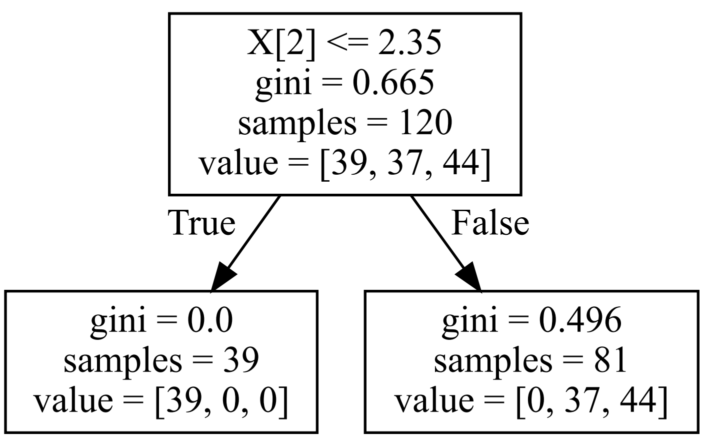
In [79]:
```
#for score
distree1_cr1 = metrics.classification_report(Y_test, predicted_distree1, targe
t_names=iris.target_names)
print(distree1_cr1)
distree1_cm1 = metrics.confusion_matrix(Y_test, predicted_distree1)
print(distree1_cm1)
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        11
  versicolor       0.00      0.00      0.00        13
   virginica       0.32      1.00      0.48         6

    accuracy                           0.57        30
   macro avg       0.44      0.67      0.49        30
weighted avg       0.43      0.57      0.46        30

[[11  0  0]
 [ 0  0 13]
 [ 0  0  6]]
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:127
2: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter t
o control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

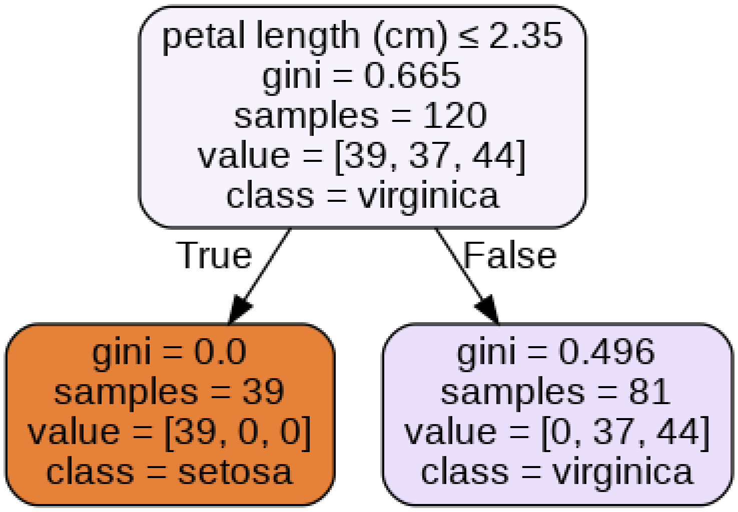
In [80]:
```
graphviz.Source(tree.export_graphviz(distree1))
```

Out[80]:



In [81]:
```
from sklearn.externals.six import StringIO
import pydotplus
from IPython.display import Image
dot_data = StringIO()
tree.export_graphviz(distree1, out_file=dot_data,
                     filled = True, rounded = True, special_characters=True, f
eature_names = iris.feature_names, class_names = iris.target_names)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[81]:



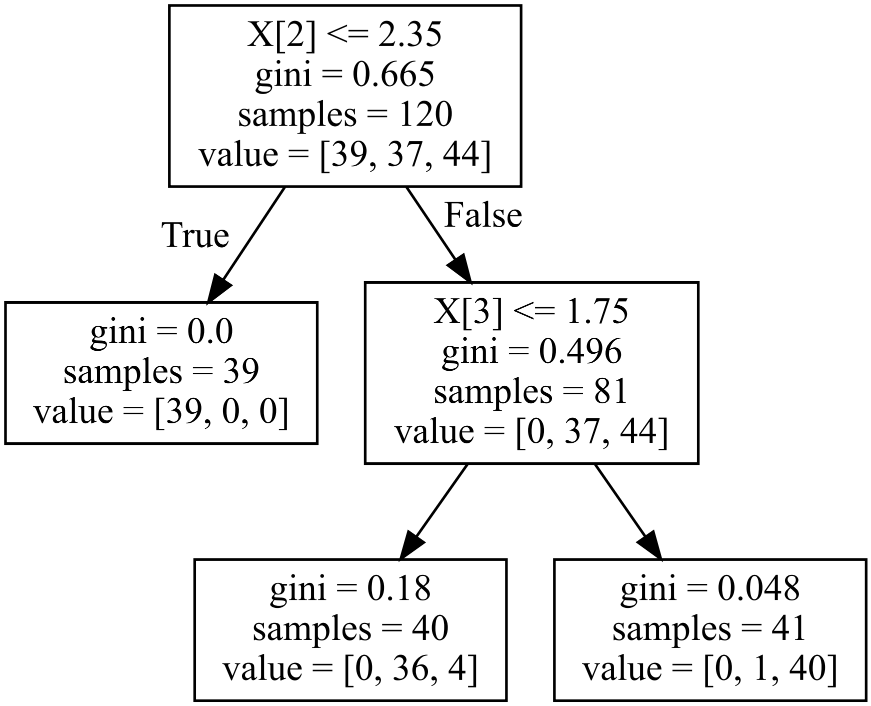

For Depth = 2

In [82]:
```
distree2 = tree.DecisionTreeClassifier(criterion='gini', max_depth=2, min_samp
les_leaf=2)
distree2.fit(X_train, Y_train)
predicted_distree2 = distree2.predict(X_test)
#for score
distree2_cr2 = metrics.classification_report(Y_test, predicted_distree2, targe
t_names=iris.target_names)
print(distree2_cr2)
distree2_cm2 = metrics.confusion_matrix(Y_test, predicted_distree2)
print(distree2_cm2)
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        11
  versicolor       0.93      1.00      0.96        13
   virginica       1.00      0.83      0.91         6

    accuracy                           0.97        30
   macro avg       0.98      0.94      0.96        30
weighted avg       0.97      0.97      0.97        30

[[11  0  0]
 [ 0 13  0]
 [ 0  1  5]]
```

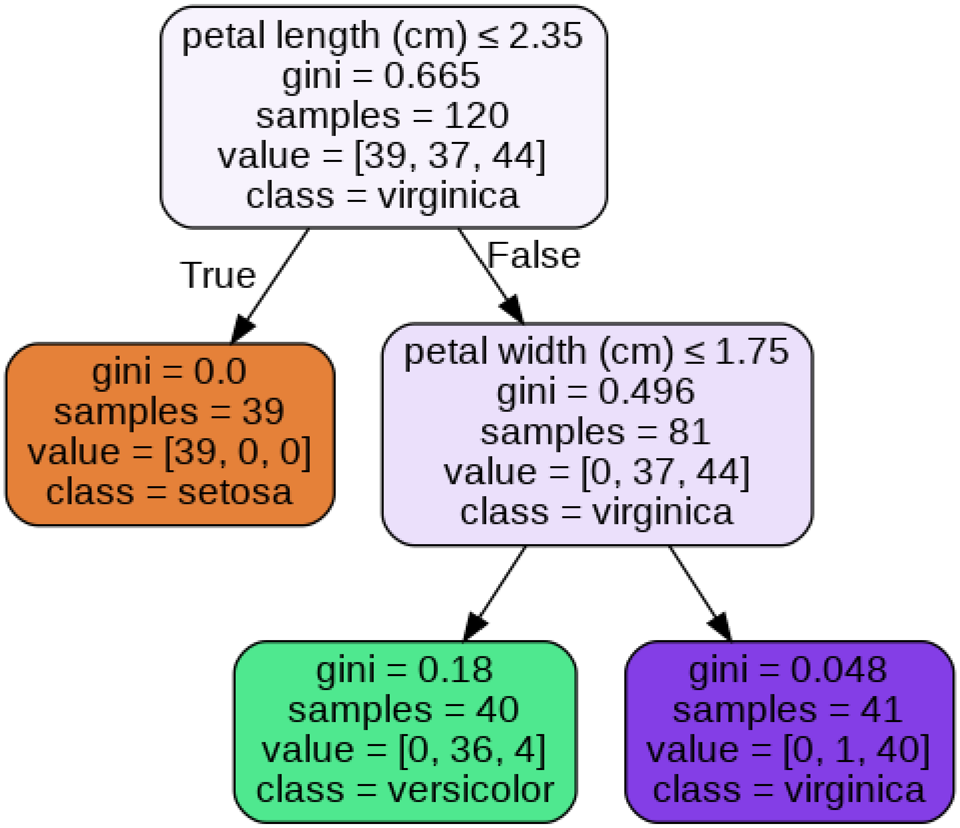
In [83]:
```
graphviz.Source(tree.export_graphviz(distree2))
```

Out[83]:



In [84]:
```
from sklearn.externals.six import StringIO
import pydotplus
from IPython.display import Image
dot_data = StringIO()
tree.export_graphviz(distree2, out_file=dot_data,
                     filled = True, rounded = True, special_characters=True, f
eature_names = iris.feature_names, class_names = iris.target_names)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[84]:



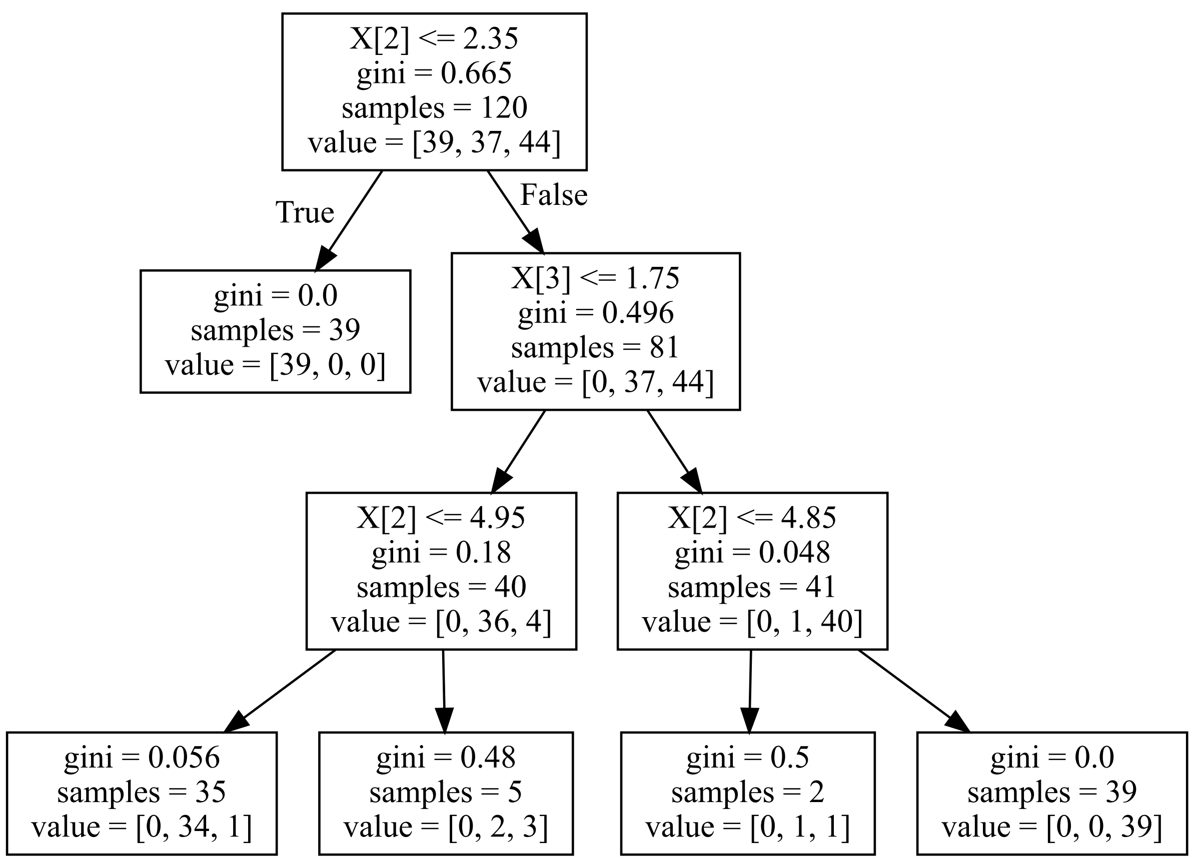

For Depth = 3

In [85]:
```
distree3 = tree.DecisionTreeClassifier(criterion='gini', max_depth=3, min_samp
les_leaf=2)
distree3.fit(X_train, Y_train)
predicted_distree3 = distree3.predict(X_test)
#for score
distree3_cr3 = metrics.classification_report(Y_test, predicted_distree3, targe
t_names=iris.target_names)
print(distree3_cr3)
distree3_cm3 = metrics.confusion_matrix(Y_test, predicted_distree3)
print(distree3_cm3)
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        11
  versicolor       0.93      1.00      0.96        13
   virginica       1.00      0.83      0.91         6

    accuracy                           0.97        30
   macro avg       0.98      0.94      0.96        30
weighted avg       0.97      0.97      0.97        30

[[11  0  0]
 [ 0 13  0]
 [ 0  1  5]]
```

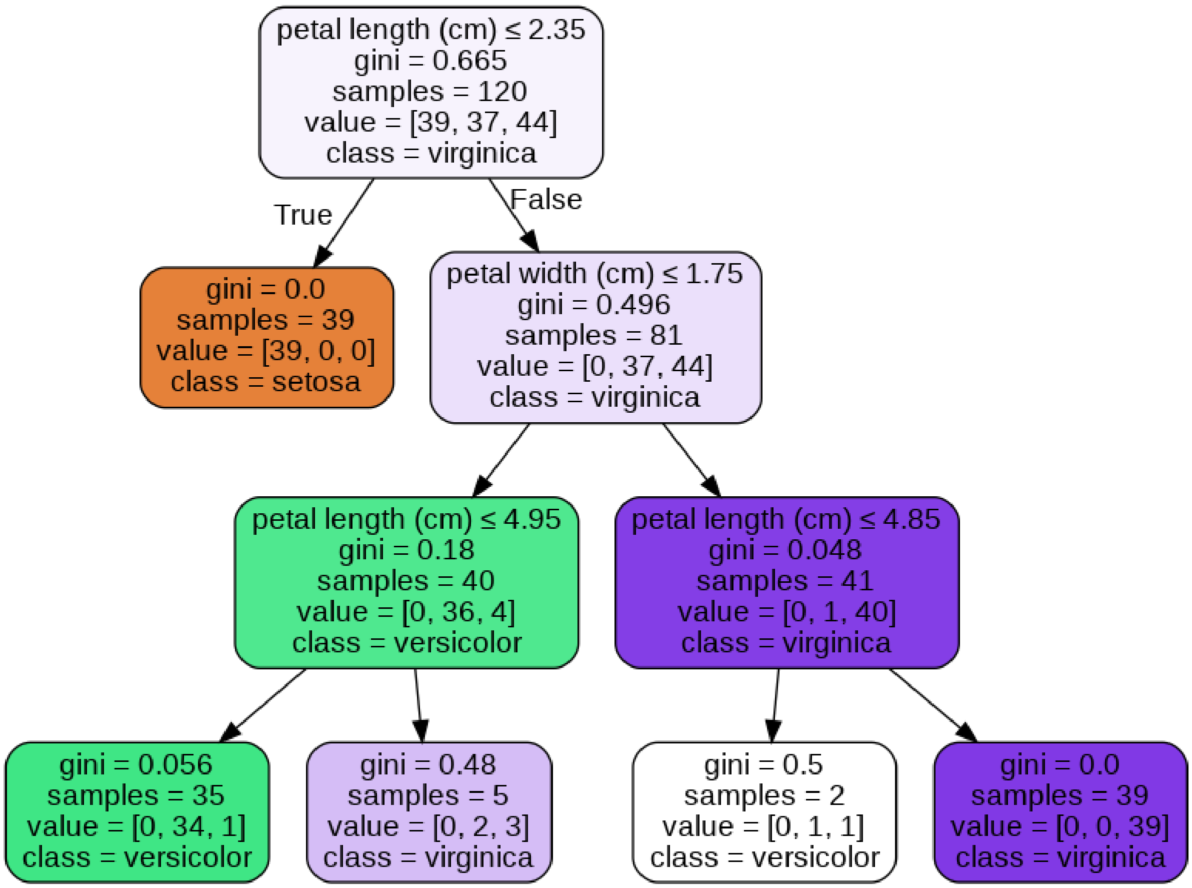
In [86]:
```
graphviz.Source(tree.export_graphviz(distree3))
```

Out[86]:



In [87]:
```
from sklearn.externals.six import StringIO
import pydotplus
from IPython.display import Image
dot_data = StringIO()
tree.export_graphviz(distree3, out_file=dot_data,
                     filled = True, rounded = True, special_characters=True, f
eature_names = iris.feature_names, class_names = iris.target_names)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[87]:



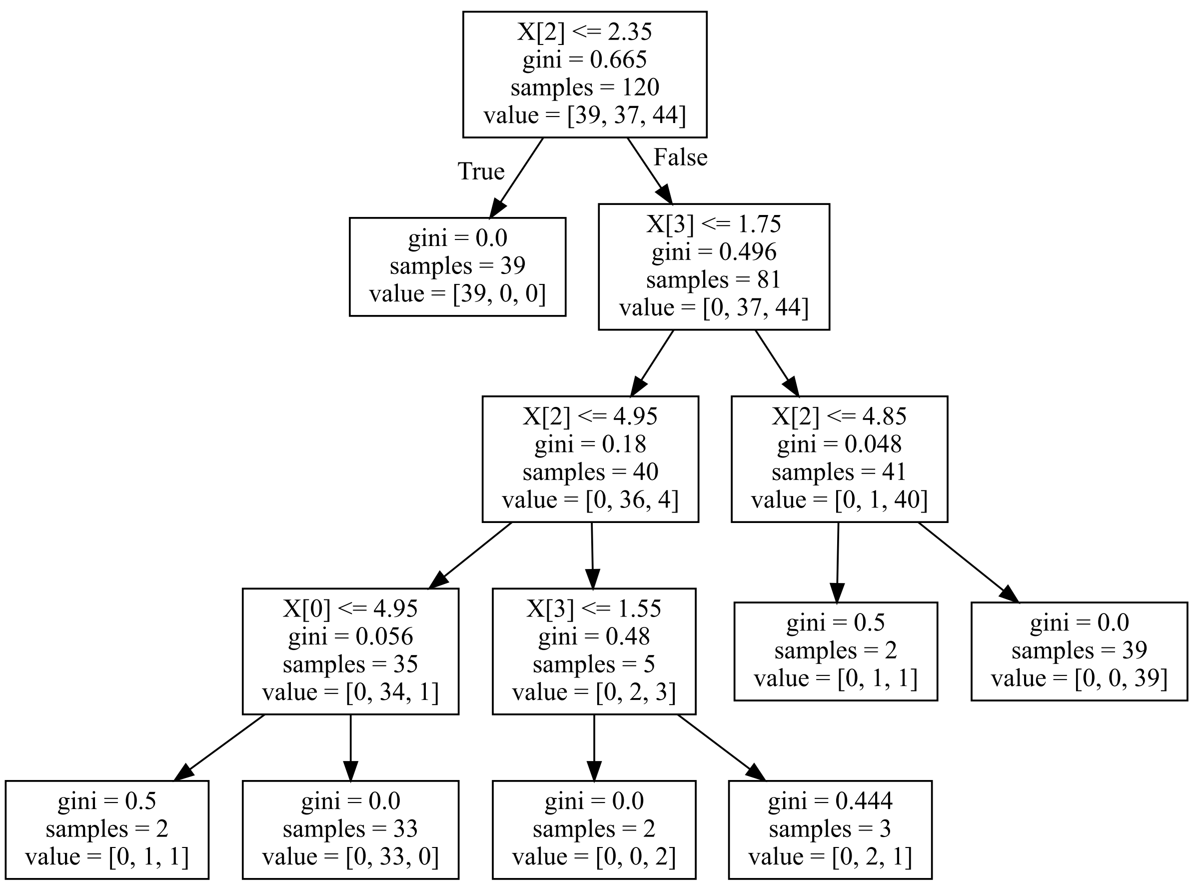

For Depth = 4

In [88]:
```
distree4 = tree.DecisionTreeClassifier(criterion='gini', max_depth=4, min_samp
les_leaf=2)
distree4.fit(X_train, Y_train)
predicted_distree4 = distree4.predict(X_test)
#for score
distree4_cr = metrics.classification_report(Y_test, predicted_distree4, target
_names=iris.target_names)
print(distree4_cr)
distree4_cm = metrics.confusion_matrix(Y_test, predicted_distree4)
print(distree4_cm)
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        11
  versicolor       0.93      1.00      0.96        13
   virginica       1.00      0.83      0.91         6

    accuracy                           0.97        30
   macro avg       0.98      0.94      0.96        30
weighted avg       0.97      0.97      0.97        30

[[11  0  0]
 [ 0 13  0]
 [ 0  1  5]]
```

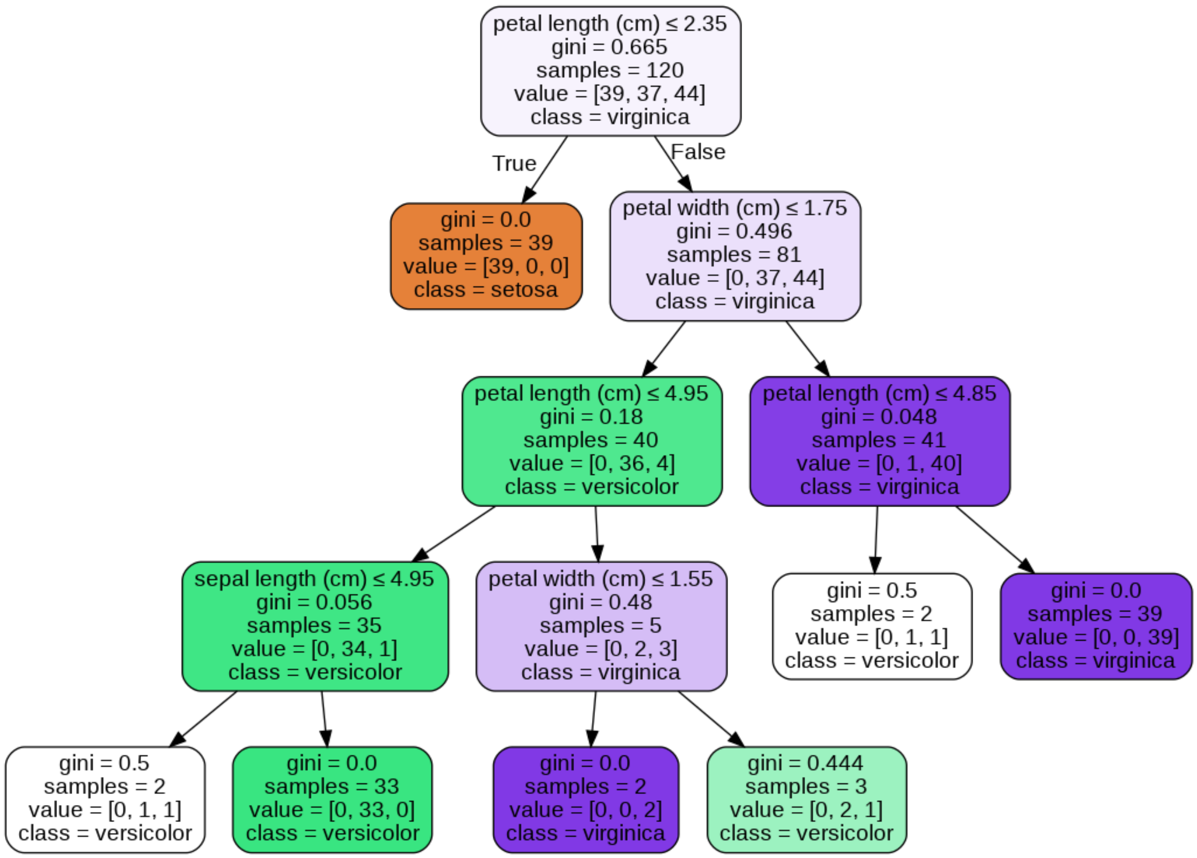
In [89]:
```
graphviz.Source(tree.export_graphviz(distree4))
```

Out[89]:



In [90]:
```
from sklearn.externals.six import StringIO
import pydotplus
from IPython.display import Image
dot_data = StringIO()
tree.export_graphviz(distree4, out_file=dot_data,
                     filled = True, rounded = True, special_characters=True, f
eature_names = iris.feature_names, class_names = iris.target_names)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[90]:



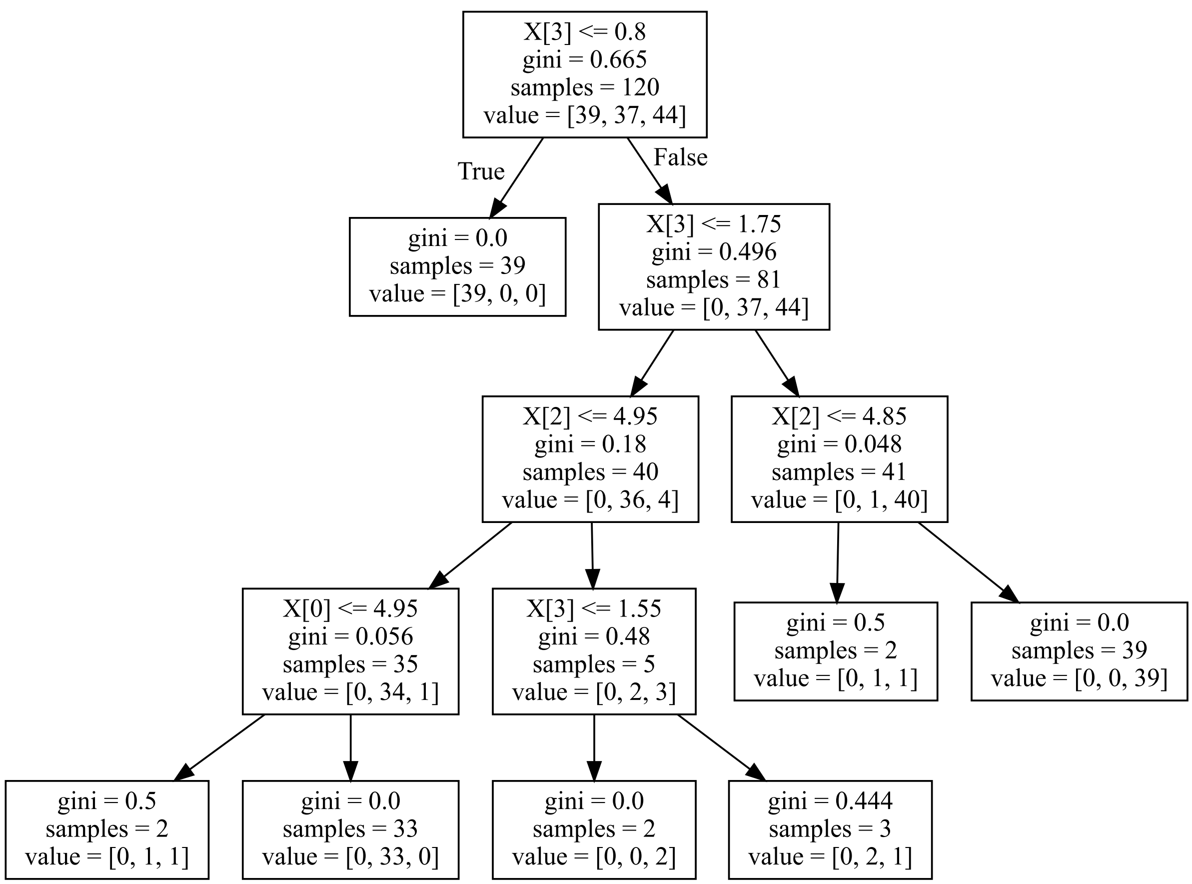

For Depth = 5

In [91]:
```
distree5 = tree.DecisionTreeClassifier(criterion='gini', max_depth=5, min_samp
les_leaf=2)
distree5.fit(X_train, Y_train)
predicted_distree5 = distree5.predict(X_test)
#for score
distree5_cr = metrics.classification_report(Y_test, predicted_distree5, target
_names=iris.target_names)
print(distree5_cr)
distree5_cm = metrics.confusion_matrix(Y_test, predicted_distree5)
print(distree5_cm)
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        11
  versicolor       0.93      1.00      0.96        13
   virginica       1.00      0.83      0.91         6

    accuracy                           0.97        30
   macro avg       0.98      0.94      0.96        30
weighted avg       0.97      0.97      0.97        30

[[11  0  0]
 [ 0 13  0]
 [ 0  1  5]]
```

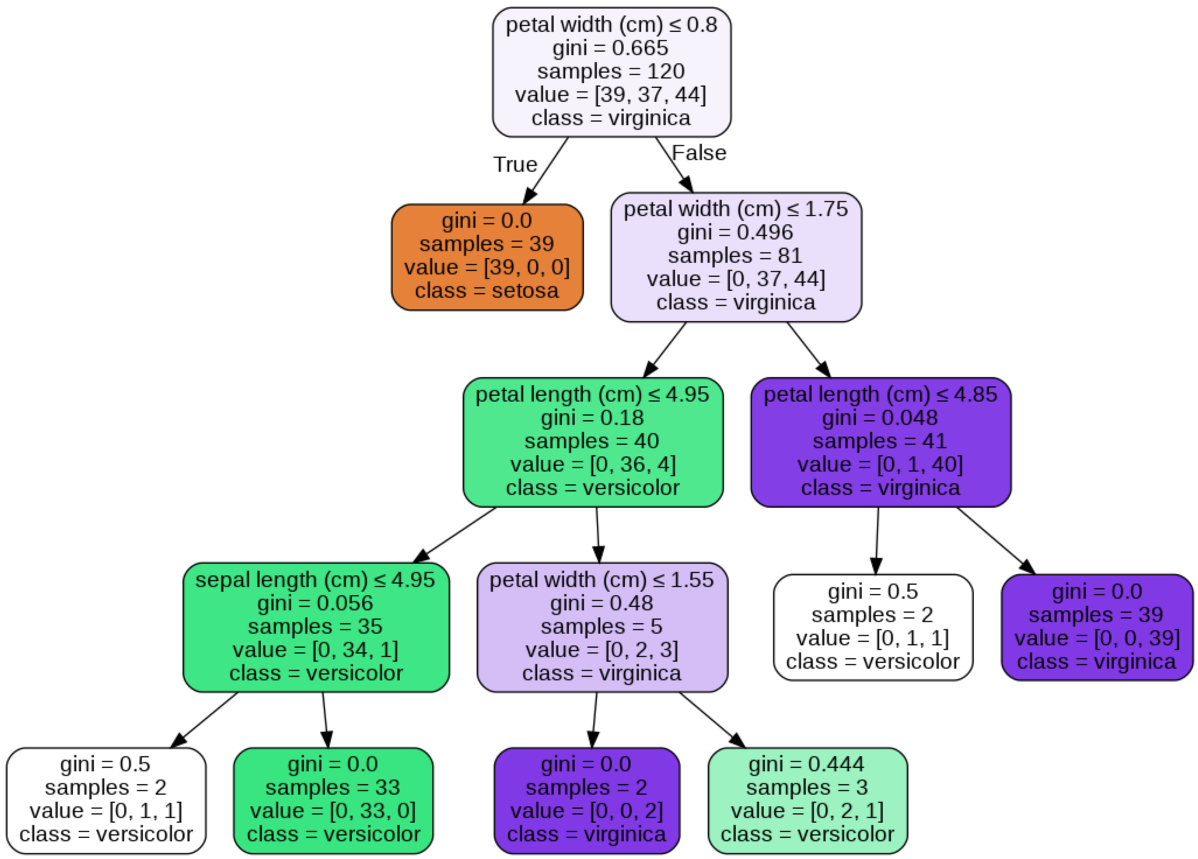
In [92]:
```
graphviz.Source(tree.export_graphviz(distree5))
```

Out[92]:



In [93]:
```
from sklearn.externals.six import StringIO
import pydotplus
from IPython.display import Image
dot_data = StringIO()
tree.export_graphviz(distree5, out_file=dot_data,
                     filled = True, rounded = True, special_characters=True, f
eature_names = iris.feature_names, class_names = iris.target_names)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[93]:




1- Depth values 2,3,4,5 has the highest recall. More the depth more the clarity.(0.97)

2- Depth =1, has the lowest precision, (0.43). This is because the tree has only one split, low precision rate indicates higher false positive values.

3- Depth 2,3,4,5 has the highest F1 scores.(0.97). F1 gives us best precision and recall values. Higher is the precision and recall, higher is the value of F1.

1- Micro-average method, sum up the individual true positives, false positives, and false negatives of the data for different sets and then apply them to get the statistics.

2- Macro-average method, calculate the average of the precision and recall of the system on different sets.

3 - Macro-average method can be used when we have to know how the entire system performs overall across the sets of data whereas, micro-average can be a useful measure when our dataset varies in size.

# PROBLEM 2

In [94]:
```
import numpy as np
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn import preprocessing
#from sklearn.preprocessing import Imputer
data = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/
breast-cancer-wisconsin/breast-cancer-wisconsin.data",
                   header = None,
                   names = ['Sample_code_number', 'Clump_Thickness', 'Uniformi
ty_of_Cell_Size', 'Uniformity_of_Cell_Shape','Marginal_Adhesion',
                            'Single_Epithelial_Cell_Size','Bare_Nuclei','Bland
_Chromatin','Normal_Nucleoli','Mitoses','Class'])
```

In [95]:
```
data.head(1)
```

Out[95]:

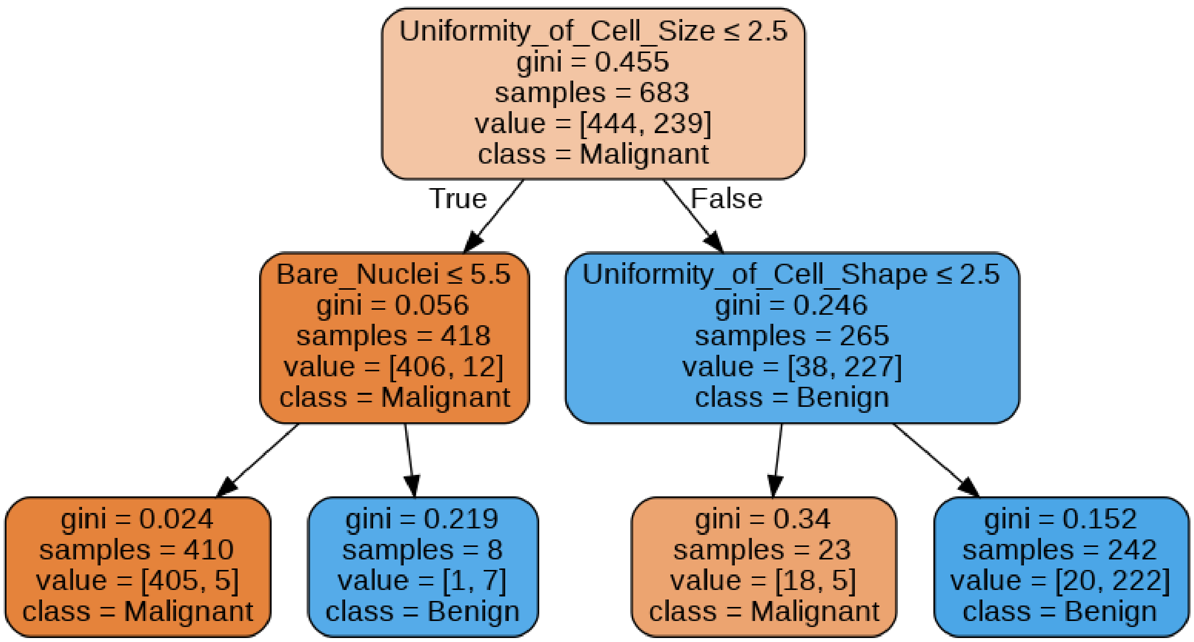
| | Sample_code_number | Clump_Thickness | Uniformity_of_Cell_Size | Uniformity_of_Cell |
|---|---|---|---|---|
| **0** | 1000025 | 5 | 1 | 1 |

In [96]:
```
data = data.replace('?',np.nan)
data = data[pd.notnull(data['Bare_Nuclei'])]
```

In [97]:
```
X = data.drop(['Class'], axis=1)
Y = data['Class']
featureName_List = list(data)[0:10]
```

In [98]:
```
clf = tree.DecisionTreeClassifier(criterion='gini', max_depth=2, min_samples_leaf=2)
clf.fit(X, Y)
from sklearn.externals.six import StringIO
import pydotplus
from IPython.display import Image
dot_data = StringIO()
tree.export_graphviz(clf, out_file=dot_data,
                     filled = True, rounded = True, special_characters=True, feature_names = featureName_List, class_names = ['Malignant', 'Benign'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[98]:




Feature selected for first split: Uniformity of cell Size.

Gini index of the first split is 1 - (444/683)^2 - (239/683)^2 = **0.455**

Entropy of the first split is -(444/683)log2(444/683) -(239/683)log2(239/683) = **0.9340**

Misclassification error of the first split is 1 - (444/683) = **0.345**

Information gain 0.9340 - (444/683)(entropy of left child=.1878) - (239/683)(entropy of right child=.5930) = **0.6044**

The decision boundary - **2.5**

# PROBLEM 3

In [164]:
```
cancer_data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-dat
abases/breast-cancer-wisconsin/wdbc.data',
                        header = None,
                        names = ["ID","diagnosis","radius_mean","texture_mean"
,"perimeter_mean","area_mean","smoothness_mean","compactness_mean","concavity_
mean","concavepnts_mean","symmetry_mean",
                                 "fractal_mean","radius_SE","texture_SE","peri
meter_SE","area_SE","smoothness_SE","compactness_SE","concavity_SE","concavept
s_SE","symmetry_SE","fractal_SE","vradiuse_worst",
                                 "texture_worst","perimeter_worst","area_wors
t","smoothness_worst","compactness_worst","concavity_worst","concave_worst","s
ymmetry_worst","fractal_worst"])
```

In [165]:
```
cancer_data.head(1)
```

Out[165]:

| | ID | diagnosis | radius_mean | texture_mean | perimeter_mean | area_mean | smoothi |
|---|---|---|---|---|---|---|---|
| **0** | 842302 | M | 17.99 | 10.38 | 122.8 | 1001.0 | 0.1184 |

In [166]:
```
cancer_data = cancer_data.replace('M', 1)
cancer_data = cancer_data.replace('B', 2)
cancer_data = cancer_data.drop(["ID"], axis=1)
X = cancer_data[['diagnosis']]
cancer_data = cancer_data.drop(['diagnosis'], axis=1)
```

In [167]:
```
from sklearn.decomposition import PCA
from sklearn import decomposition

pca = PCA()
X_train = pca.fit_transform(cancer_data)
explained_variance = pca.explained_variance_ratio_
print("Varinace of the PCA", explained_variance)
```

```
Varinace of the PCA [9.82044672e-01 1.61764899e-02 1.55751075e-03 1.20931964e
-04
 8.82724536e-05 6.64883951e-06 4.01713682e-06 8.22017197e-07
 3.44135279e-07 1.86018721e-07 6.99473205e-08 1.65908880e-08
 6.99641650e-09 4.78318306e-09 2.93549214e-09 1.41684927e-09
 8.29577731e-10 5.20405883e-10 4.08463983e-10 3.63313378e-10
 1.72849737e-10 1.27487508e-10 7.72682973e-11 6.28357718e-11
 3.57302295e-11 2.76396041e-11 8.14452259e-12 6.30211541e-12
 4.43666945e-12 1.55344680e-12]
```

In [168]:
```
#before PCA
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(cancer_dat
a, X, test_size=0.2)

distree = tree.DecisionTreeClassifier(max_depth=2, min_samples_leaf=2, min_sam
ples_split=5)
distree = distree.fit(X_train, Y_train)
predicted= distree.predict(X_test)
#for score
cancer_data_distree_cr = metrics.classification_report(Y_test, predicted)
print(cancer_data_distree_cr)
cancer_data_distree_cm = metrics.confusion_matrix(Y_test, predicted)
print(cancer_data_distree_cm)
```

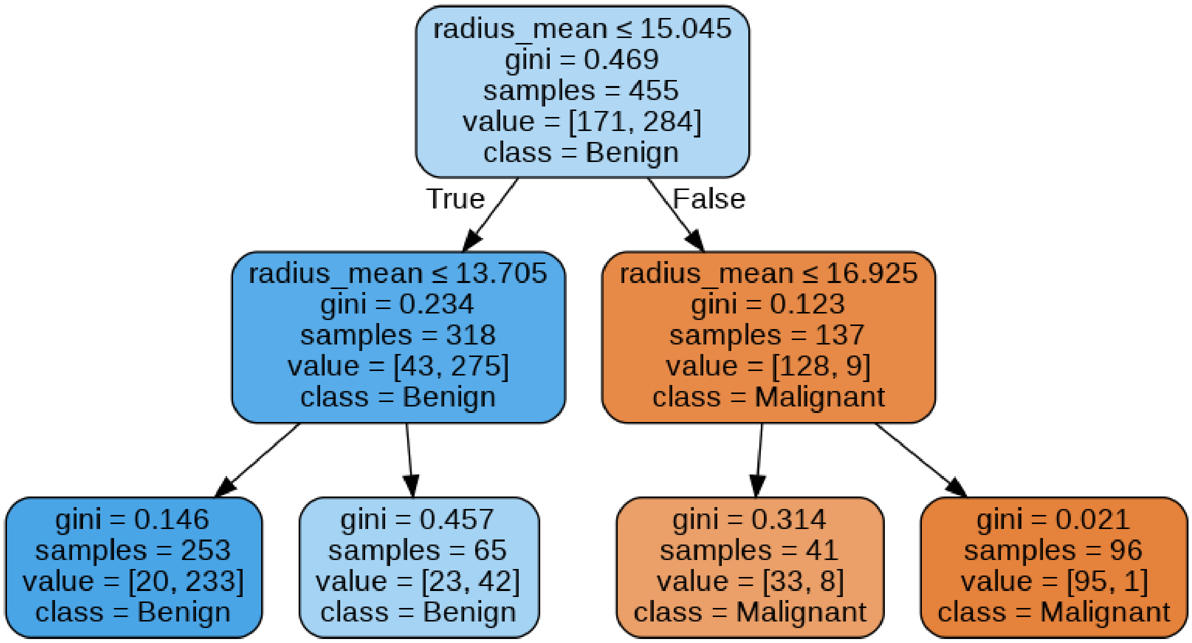
```
              precision    recall  f1-score   support

           1       0.86      0.90      0.88        40
           2       0.94      0.92      0.93        74

    accuracy                           0.91       114
   macro avg       0.90      0.91      0.90       114
weighted avg       0.91      0.91      0.91       114

[[36  4]
 [ 6 68]]
```

In [169]:
```
# with PCA with 1st component
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(cancer_dat
a[['radius_mean']], X, test_size=0.2)
distree = tree.DecisionTreeClassifier(max_depth=2, min_samples_leaf=2, min_sam
ples_split=5)
distree.fit(X_train, Y_train)
predicted= distree.predict(X_test)
#for score
cancer_data_distree_cr = metrics.classification_report(Y_test, predicted)
print(cancer_data_distree_cr)
cancer_data_distree_cm = metrics.confusion_matrix(Y_test, predicted)
print(cancer_data_distree_cm)
```

```
              precision    recall  f1-score   support

           1       0.94      0.80      0.87        41
           2       0.90      0.97      0.93        73

    accuracy                           0.91       114
   macro avg       0.92      0.89      0.90       114
weighted avg       0.91      0.91      0.91       114

[[33  8]
 [ 2 71]]
```

In [170]:
```
dot_data = StringIO()
tree.export_graphviz(distree, out_file=dot_data,
                     filled = True, rounded = True, special_characters=True, f
eature_names = ['radius_mean'], class_names = ['Malignant', 'Benign'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[170]:

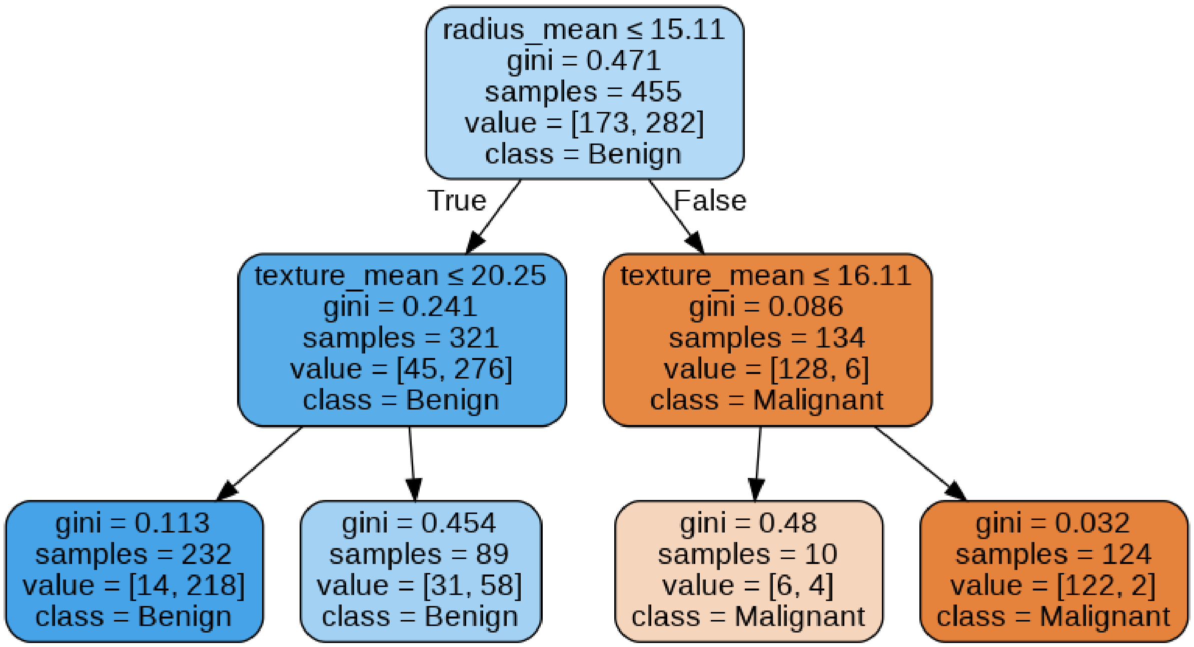



In [171]:
```
# with PCA with 1st and 2nd component
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(cancer_dat
a[['radius_mean']].join(cancer_data[['texture_mean']]), X, test_size=0.2)
distree = tree.DecisionTreeClassifier(max_depth=2, min_samples_leaf=2, min_sam
ples_split=5)
distree.fit(X_train, Y_train)
predicted= distree.predict(X_test)
#for score
cancer_data_distree_cr = metrics.classification_report(Y_test, predicted)
print(cancer_data_distree_cr)
cancer_data_distree_cm = metrics.confusion_matrix(Y_test, predicted)
print(cancer_data_distree_cm)
```

```
              precision    recall  f1-score   support

           1       0.88      0.74      0.81        39
           2       0.88      0.95      0.91        75

    accuracy                           0.88       114
   macro avg       0.88      0.85      0.86       114
weighted avg       0.88      0.88      0.87       114

[[29 10]
 [ 4 71]]
```

In [172]:
```
dot_data = StringIO()
tree.export_graphviz(distree, out_file=dot_data,
                     filled = True, rounded = True, special_characters=True, f
eature_names = ['radius_mean', 'texture_mean'], class_names = ['Malignant', 'B
enign'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[172]:



We can see the difference in test & score tables of original data and the data on which PCA is performed, the F1 score, precision and recall values have increased after PCA is carried out. So, PCA –based single factor model is better.

Calculations:

Before PCA:

[[36 4] [ 6 68]]

FP: 4, TP: 68, TPR: 68/(68+6) = 0.9189, FPR: 4/(4+36) = 0.10

PCA with 1st component:

[[33 8] [ 2 71]]

FP: 8, TP: 71, TPR: 71/(71+2) = 0.9726, FPR: 8/(8+33) = 0.195

PCA with 1st and 2nd component:

[[29 10] [ 4 71]]

FP: 10, TP: 71, TPR: 71/(71+4) = 0.9466, FPR: 10/(10+29) = 0.2564

Yes, using continuous data is beneficial in this model. PCA creates variables that are linear combinations of the original variables. Our aim is to find clusters of data. This is the reason we perform Principal Component Analysis. And if the values are continuous, the analysis is better.

# PROBLEM 4

In [177]:

```
x1 = np.random.normal(5, 2, 2000)
x2 = np.random.normal(-5, 2, 2000)
p1 = np.repeat('p1', 2000)
p2 = np.repeat('p2', 2000)
data_frame1 = pd.DataFrame(dict(zip(['x','y'], [x1,p1])))
data_frame2 = pd.DataFrame(dict(zip(['x','y'], [x2,p2])))

combo_df = pd.concat([data_frame1,data_frame2])

Y = combo_df.y
X = combo_df.x.values.reshape(-1,1)
features = list(combo_df.y)
```

In [180]:
```
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y, test
_size=0.2, random_state=0)
clf = tree.DecisionTreeClassifier(max_depth=2)
clf.fit(X_train, Y_train)
Y_predicted= clf.predict(X_test)

cr = metrics.classification_report(Y_test, Y_predicted)
print(cr)
```

```
              precision    recall  f1-score   support

          p1       1.00      0.98      0.99       407
          p2       0.98      1.00      0.99       393

    accuracy                           0.99       800
   macro avg       0.99      0.99      0.99       800
weighted avg       0.99      0.99      0.99       800
```

In [181]:
```
clf.tree_.threshold[0]
```

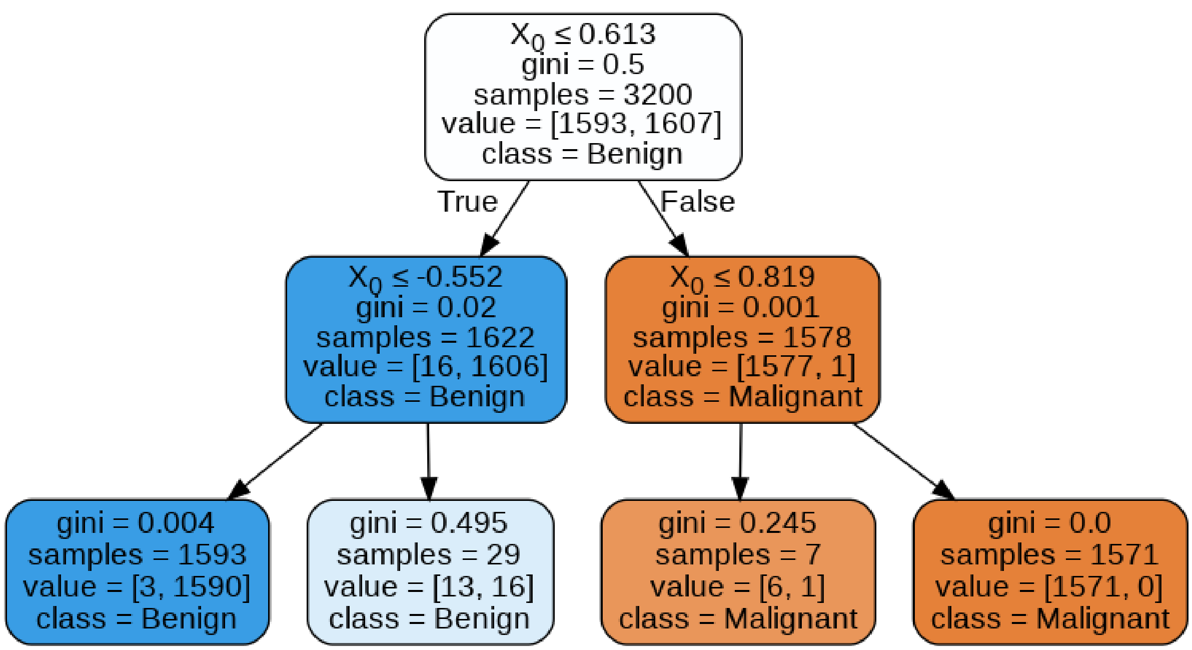
Out[181]: 0.6133889555931091

In [182]:
```
clf.tree_.feature[0]
```

Out[182]: 0

In [183]:
```
dot_data = StringIO()
tree.export_graphviz(clf, out_file=dot_data,
                     filled = True, rounded = True, special_characters=True, c
lass_names = ['Malignant', 'Benign'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[183]:



Threshold value depends on the measure we have built the decision tree.

Here, we have chosen Gini Index to decide the split points.

Therefore to calculate empirical distribution value, the threshold of feature value must be exceeded.