# Project 2 Report

First and Last name

CS458 or CS658

# P2-1. Decision Tree

**(a) Develop a decision tree based classifier to classify the 3 different types of Iris (Setosa, Versicolour, and Virginica).**

In [2]:

```
# Codes for P2-1(a)
from sklearn import (datasets, tree, model_selection, metrics)
import numpy as np
import pydotplus
from IPython.display import Image

iris = datasets.load_iris()

training_data = []
training_target = []

test_data = []
test_target = []

for targets in range(max(iris.target) + 1):
    index = np.where(iris.target == targets)[0][0]
    test_data.extend(iris.data[index:index + 10])
    test_target.extend([targets] * 10)

    training_data.extend(iris.data[index + 10:index + 50])
    training_target.extend([targets] * 40)

clf = tree.DecisionTreeClassifier(
    max_depth=3, min_samples_leaf=2, random_state=0, max_features="auto")
clf.fit(training_data, training_target)

classified_target = clf.predict(test_data)

score = model_selection.cross_val_score(clf, test_data, test_target, cv=5)
```

Discuss how you use 5-fold cross validation to train the classifier here.

I used Cross Validation to to validate our model. I split the entire data randomly into 5 folds then used the model using the K - 1 folds and validated the model with the remaining K'th fold by returning the score. The process was repeated and then averaged

**(b) Optimize the parameters of your decision tree to maximize the classification accuracy. Show the confusion matrix of your decision tree. Plot your decision tree.**

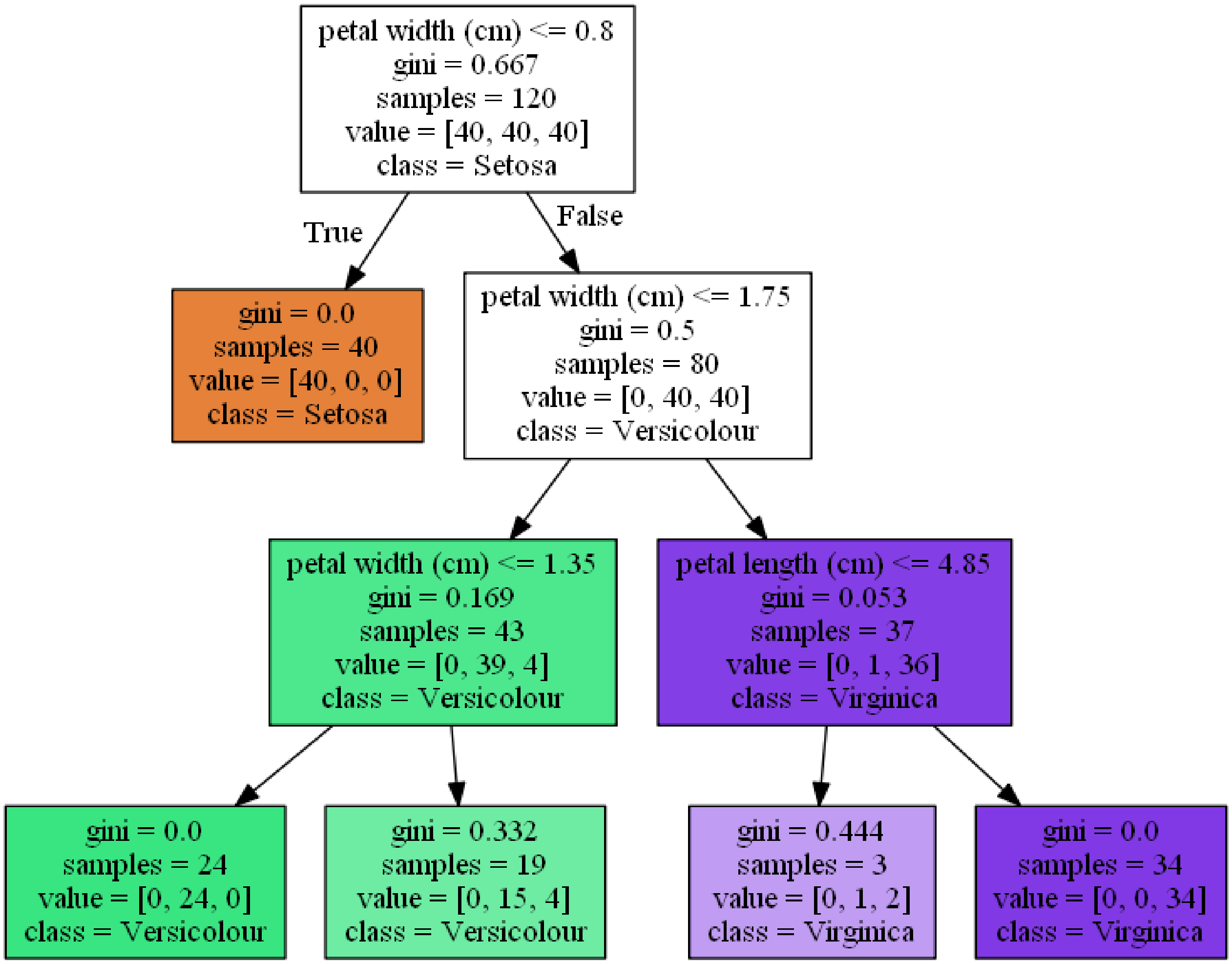
In [3]:

```
# Codes for P2-1(b)
print('Print your classification accuracy, confusion matrix and plot your decision tree
print(clf.score(test_data, test_target))
print (metrics.confusion_matrix(test_target, classified_target))
```

```
dot_data = tree.export_graphviz(clf, feature_names=iris['feature_names'], class_names=[
                                out_file=None)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

```
Print your classification accuracy, confusion matrix and plot your decision tree.
0.9666666666666667
[[10  0  0]
 [ 0 10  0]
 [ 0  1  9]]
```

Out[3]:



Discuss how you optimize the parameters of your decision tree here.

I used Gini Index to optimize the parameters of the decision tree. This shows how much of the training data in a particular region belongs to a single class.

# P2-2. Model Overfitting

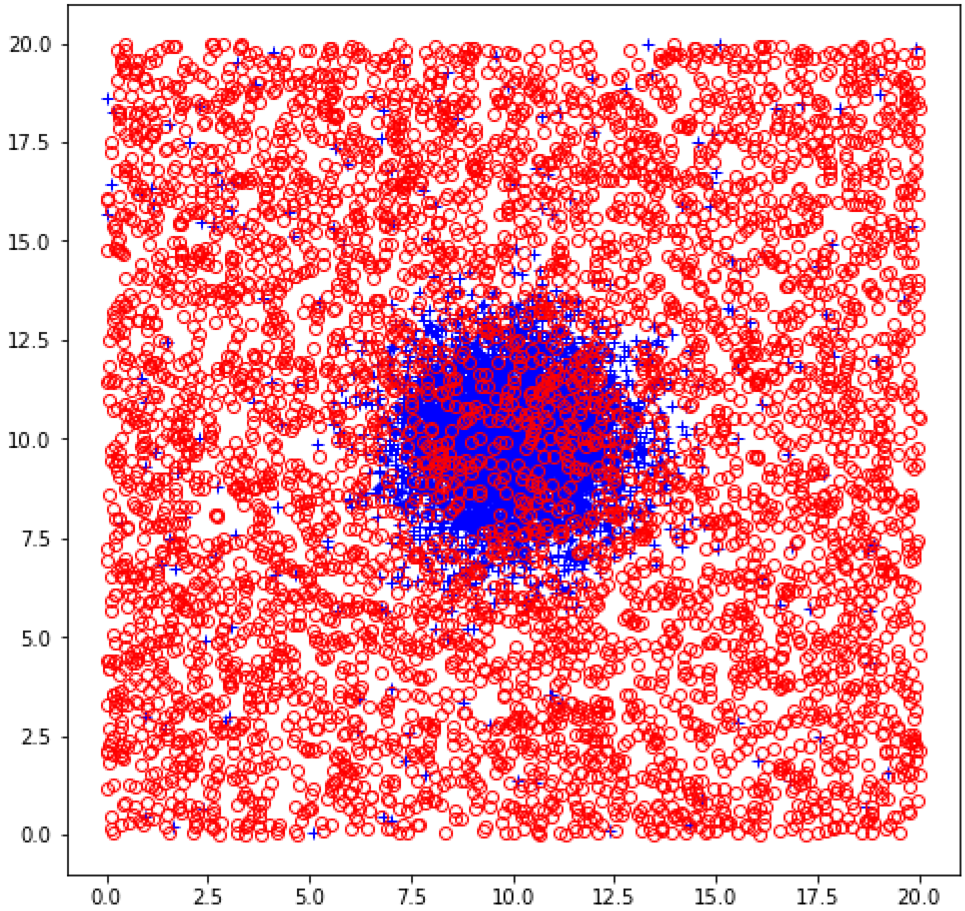
**(a) Generate the dataset as in slide 56 in Chapter 3**

In [8]:

```
# Codes for P2-2(a)
print('Plot your dataset')

# Codes for P2-2(a)
import numpy as np
import matplotlib.pyplot as plt
from numpy.random import random
```

```
N = 5200
mean = [10,10]
cov = [[2,0],[0,2]]
np.random.seed(0)
X = np.random.multivariate_normal(mean, cov,N-200)
X = np.concatenate((X, np.random.uniform(0 ,20, (200,2))))
X = np.concatenate((X, np.random.uniform(0 ,20, (N,2))))
Y = np.concatenate((np.ones(N),np.zeros(N)))

plt.figure(figsize=(8,8))

plt.plot(X[:N,0], X[:N,1], 'b+', X[N:,0], X[N:,1], 'ro', fillstyle= 'none')
plt.show()
plt.clf()
```

```
Plot your dataset
```



```
<Figure size 432x288 with 0 Axes>
```

**(b) Randomly select 10% of the data as test dataset and the remaining 90% of the data as training dataset. Train decision trees by increasing the number of nodes of the decision trees until the training error becomes 0. Plot the training errors and the testing errors under different numbers of nodes and explain the model underfitting and model overfitting.**

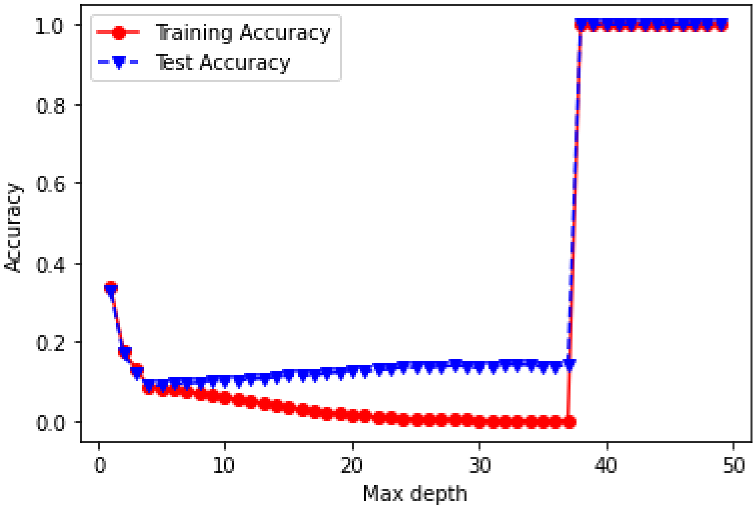
In [11]:

```
# Codes for P2-2(b)
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.1, random_state=1

from sklearn import tree
from sklearn.metrics import accuracy_score
```

```
print('Plot the training errors and the testing errors under different numbers of nodes
print(X.shape)
maxd = range(1,50)
train_acc = np.zeros(len(maxd))
test_acc = np.zeros(len(maxd))
index = 0
train_err = 100.0
while(train_err!=0.0):
    clf = tree.DecisionTreeClassifier(max_depth=maxd[index])
    clf = clf.fit(X_train, Y_train)
    Y_predTrain = clf.predict(X_train)
    Y_predTest = clf.predict(X_test)
    train_acc[index] = accuracy_score(Y_train, Y_predTrain)
    test_acc[index] = accuracy_score(Y_test, Y_predTest)
    train_err=1-train_acc[index]
    index += 1
    print(index)
    print (train_err)

plt.plot(maxd,1-train_acc,'ro-',maxd,1-test_acc,'bv--')
plt.legend(['Training Accuracy','Test Accuracy'])
plt.xlabel('Max depth')
plt.ylabel('Accuracy')
plt.show()
```

```
Plot the training errors and the testing errors under different numbers of nodes
(10400, 2)
1
0.3372863247863248
2
0.1759615384615385
3
0.13130341880341878
4
0.08643162393162396
5
0.08247863247863252
6
0.07905982905982911
7
0.07382478632478628
8
0.07008547008547006
9
0.0657051282051282
10
0.06004273504273505
11
0.05491452991452994
12
0.050854700854700896
13
0.04561965811965807
14
0.04049145299145296
15
0.03536324786324785
16
0.02991452991452992
17
0.025854700854700874
18
0.02147435897435901
```

```
19
0.01752136752136757
20
0.014957264957264904
21
0.01239316239316235
22
0.010897435897435859
23
0.008653846153846123
24
0.006944444444444442
25
0.005555555555555536
26
0.004273504273504258
27
0.0036324786324786196
28
0.002991452991452981
29
0.002136752136752129
30
0.0017094017094017033
31
0.001388888888888884
32
0.001175213675213671
33
0.0008547008547008517
34
0.0005341880341880323
35
0.0003205128205128194
36
0.00021367521367521292
37
0.0
```



Explain the model underfitting and model overfitting here.

Overfitting occured here because the production of the anaylisis corresponded too closley to the particular set of data. Underfitting occurs when a model cannot capture the underlying structure of data. As the max depth increases, the test accuracy increases causing overfitting. The training accuracy decreases as the max depth increases until a threshold is hit where enough data is obtained to achieve a higher level of accuracy.

# P2-3. Text Documents Classification

**(a) Load the following 4 categories from the 20 newsgroups dataset: categories = ['rec.autos', 'talk.religion.misc', 'comp.graphics', 'sci.space']. Print the number of documents in the training dataset and the test dataset. Print the number of attributes in the training dataset.**

In [14]:

```
# Codes for P2-3(a)
print('Print the number of documents in the training dataset and the test dataset. Prin

from sklearn.datasets import fetch_20newsgroups

categories = ['rec.autos','talk.religion.misc', 'comp.graphics', 'sci.space']

training = fetch_20newsgroups(
    subset="train",
    categories=categories
)
test = fetch_20newsgroups(
    subset="test",
    categories=categories
)

print (len(training.data), len(test.data))
print (training.target_names)
```

```
Print the number of documents in the training dataset and the test dataset. Print the nu
mber of attributes in the training dataset.
2148 1430
['comp.graphics', 'rec.autos', 'sci.space', 'talk.religion.misc']
```

**(b) Optimize the parameters of your decision tree to maximize the classification accuracy. Show the confusion matrix of your decision tree.**

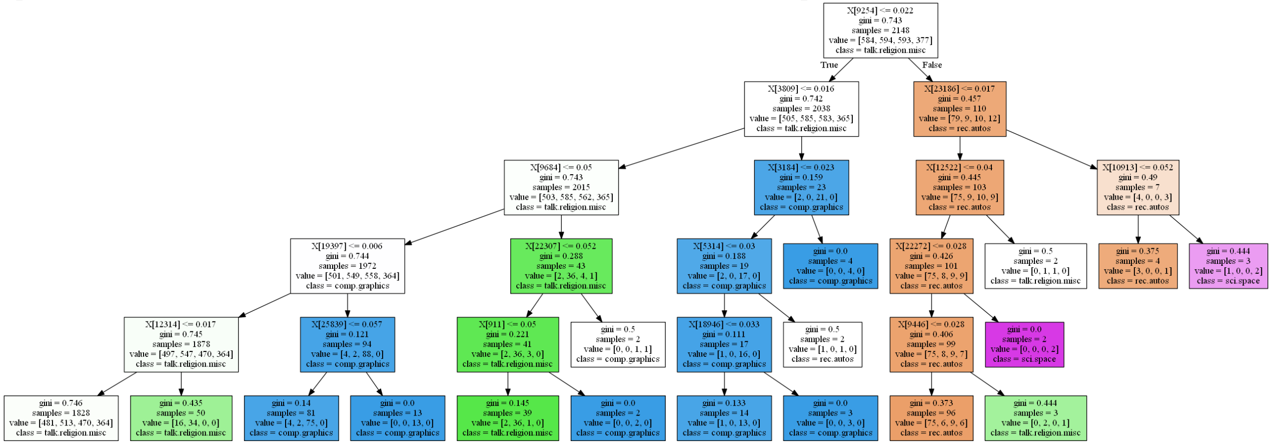
In [15]:

```
# Codes for P2-3(b)
print('Print your classification accuracy, confusion matrix.')


from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import (datasets, tree, model_selection, metrics)
import numpy as np
import pydotplus
from IPython.display import Image

categories = ['rec.autos','talk.religion.misc', 'comp.graphics', 'sci.space']

training = fetch_20newsgroups(
    subset="train",
    categories=categories
)
test = fetch_20newsgroups(
    subset="test",
    categories=categories
)
```

```
y_test, y_train = test.target, training.target

vector = TfidfVectorizer(sublinear_tf=True, max_df=0.5, stop_words='english')

X_test, X_train = vector.fit_transform(test.data), vector.fit_transform(training.data)

classifier = tree.DecisionTreeClassifier(
    max_depth=5, min_samples_leaf=2, random_state=0, max_features="auto")
classifier = classifier.fit(X_train, y_train)

score = model_selection.cross_val_score(
    classifier, X_test, y_test, cv=5)


print(score)

dot_data = tree.export_graphviz(classifier, class_names=categories, filled=True,
                                out_file=None)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

```
Print your classification accuracy, confusion matrix.
[0.34615385 0.35664336 0.43706294 0.34615385 0.36363636]
```

Out[15]:



Discuss how you optimize the parameters of your decision tree here.

I used the GINI index to optimize the parameters of my decision tree. I used the testing data to generate the confusion matrix and aid the building of the tree. I optimized the hyper parameters and itterated through to find the best combination of parameters.