In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [3]:
```
df=pd.read_csv('testing.csv')
```

In [4]:
```
df.head()
```

Out[4]:

	class	GLCM_pan	Mean_Green	Mean_Red	Mean_NIR	SD_pan
0	n	109.828571	183.700000	82.950000	251.750000	16.079412
1	n	130.284483	212.637931	96.896552	482.396552	21.210295
2	n	131.386555	185.466667	85.466667	419.666667	13.339998
3	n	141.345098	180.875000	81.500000	348.062500	18.213577
4	w	121.383408	218.357143	112.017857	426.607143	19.083196

In [5]:
```
cl={'n':1,'w':0}
```

In [6]:
```
df['class']=[cl[x] for x in df['class']]
```

In [7]:
```
df.head()
```

Out[7]:

	class	GLCM_pan	Mean_Green	Mean_Red	Mean_NIR	SD_pan
0	1	109.828571	183.700000	82.950000	251.750000	16.079412
1	1	130.284483	212.637931	96.896552	482.396552	21.210295

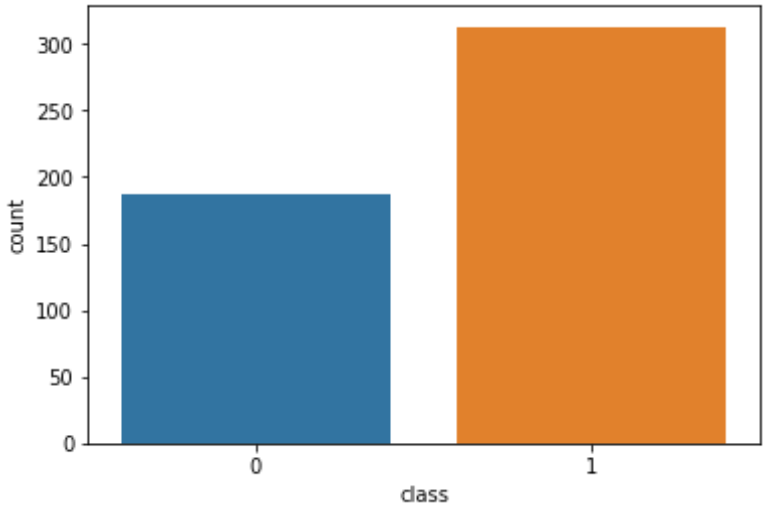
	class	GLCM_pan	Mean_Green	Mean_Red	Mean_NIR	SD_pan
2	1	131.386555	185.466667	85.466667	419.666667	13.339998
3	1	141.345098	180.875000	81.500000	348.062500	18.213577
4	0	121.383408	218.357143	112.017857	426.607143	19.083196

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 6 columns):
class          500 non-null int64
GLCM_pan       500 non-null float64
Mean_Green     500 non-null float64
Mean_Red       500 non-null float64
Mean_NIR       500 non-null float64
SD_pan         500 non-null float64
dtypes: float64(5), int64(1)
memory usage: 23.5 KB
```

In [9]:
```
sns.countplot(x='class',data=df)
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x13c4cf699e8>
```



In [10]:
```
df[df['GLCM_pan']==df.GLCM_pan.max()]
```

Out[10]:

	class	GLCM_pan	Mean_Green	Mean_Red	Mean_NIR	SD_pan
398	1	167.944444	208.25	96.375	711.75	17.200563

In [11]:
```
df.columns
```

Out[11]:
```
Index(['class', 'GLCM_pan', 'Mean_Green', 'Mean_Red', 'Mean_NIR', 'SD_p
an'], dtype='object')
```

In [12]:
```
X=df.drop('class',axis=1)
```

In [13]:
```
y=df['class']
```

In [14]:
```
from sklearn.cross_validation import train_test_split
```

```
C:\Users\Dell\Anaconda3\lib\site-packages\sklearn\cross_validation.py:4
1: DeprecationWarning: This module was deprecated in version 0.18 in fa
vor of the model_selection module into which all the refactored classes
```

```
and functions are moved. Also note that the interface of the new CV ite
rators are different from that of this module. This module will be remo
ved in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [15]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4
)
```

In [16]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [17]:
```
dtc=DecisionTreeClassifier()
```

In [18]:
```
dtc.fit(X_train,y_train)
```

Out[18]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=N
one,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=N
one,
            splitter='best')
```

In [19]:
```
predict=dtc.predict(X_test)
```

In [20]:
```
from sklearn.metrics import classification_report,confusion_matrix
```

In [21]:
```
print(classification_report(y_test,predict))
print('\n')
print(confusion_matrix(y_test,predict))
```

```
             precision    recall  f1-score   support

          0       0.84      0.83      0.84        78
          1       0.89      0.90      0.90       122

avg / total       0.87      0.88      0.87       200
```

```
[[ 65  13]
 [ 12 110]]
```

In [22]:
```
# now try with random forest
```

In [23]:
```
from sklearn.ensemble import RandomForestClassifier
```

```
C:\Users\Dell\Anaconda3\lib\site-packages\sklearn\ensemble\weight_boost
ing.py:29: DeprecationWarning: numpy.core.umath_tests is an internal Nu
mPy module and should not be imported. It will be removed in a future N
umPy release.
  from numpy.core.umath_tests import inner1d
```

In [24]:
```
rfc=RandomForestClassifier(n_estimators=300) # because we have large da
ta
```

In [25]:
```
rfc.fit(X_train,y_train)
```

Out[25]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gi
ni',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=300, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [26]:
```
predictions=rfc.predict(X_test)
```

In [27]:
```
print(classification_report(y_test,predictions))
print('\n')
print(confusion_matrix(y_test,predictions))
```

```
              precision    recall  f1-score   support
```

```
                   0       0.89      0.87      0.88        78
                   1       0.92      0.93      0.93       122

         avg / total       0.91      0.91      0.91       200



[[ 68  10]
 [  8 114]]
```

In [31]:
```
#here random forest does very well job in fitting
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: