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_train=pd.read_csv('testing.csv')
```

In [4]:
```
df_train.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]:
```
df_train.describe()
```

Out[5]:

	GLCM_pan	Mean_Green	Mean_Red	Mean_NIR	SD_pan
count	500.000000	500.000000	500.000000	500.000000	500.000000
mean	127.065977	209.767564	107.739215	453.734870	20.641288
std	10.667542	78.677763	71.773037	156.198323	6.757322
min	81.125000	117.210526	50.578947	144.875817	5.772400
25%	119.978475	188.892662	85.511304	341.588922	15.853416

	GLCM_pan	Mean_Green	Mean_Red	Mean_NIR	SD_pan
50%	127.532191	203.626923	99.828421	443.719444	20.028992
75%	133.799711	218.965116	118.054555	542.959928	24.121108
max	167.944444	1848.916667	1594.583333	1597.333333	62.396581

In [6]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 6 columns):
class          500 non-null object
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), object(1)
memory usage: 23.5+ KB
```

In [8]:
```
clas={'n':1,'w':0}
```

In [9]:
```
df_train['class']=[clas[item] for item in df_train['class']]
```

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

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

In [12]:
```
X_train=df_train[[ 'GLCM_pan', 'Mean_Green', 'Mean_Red', 'Mean_NIR', 'S
D_pan']]
y_train=df_train['class']
```

In [21]:
```
from sklearn.linear_model import LogisticRegression
```

In [22]:
```
lr=LogisticRegression()
```

In [23]:
```
lr.fit(X_train,y_train)
```

Out[23]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=
True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=
1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.00
01,
          verbose=0, warm_start=False)
```

In [13]:
```
#now its testing time :-)
```

In [14]:
```
df_tes=pd.read_csv('testing.csv')
```

In [15]:
```
df_tes.head()
```

Out[15]:

	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 [16]:
```
df_tes['class']=[clas[x] for x in df_tes['class']]
```

In [17]:
```
df_tes
```

Out[17]:

	class	GLCM_pan	Mean_Green	Mean_Red	Mean_NIR	SD_pan
0	1	109.828571	183.700000	82.950000	251.750000	16.079412

	class	GLCM_pan	Mean_Green	Mean_Red	Mean_NIR	SD_pan
1	1	130.284483	212.637931	96.896552	482.396552	21.210295
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
5	1	122.757576	205.960000	86.760000	407.680000	17.823580
6	0	124.010204	215.594595	117.027027	477.297297	24.574628
7	0	125.608407	209.649123	121.228070	443.280702	25.757314
8	1	107.745833	204.133333	83.466667	479.866667	22.956965
9	1	140.203922	188.593750	84.437500	457.250000	30.193759
10	1	122.792453	199.740741	108.481482	384.740741	23.018906
11	1	117.427907	191.185185	86.555556	429.703704	24.765567
12	1	130.966357	256.352113	166.683099	506.964789	18.462242
13	1	132.678821	259.126984	170.341270	519.912698	23.550458
14	1	132.678821	259.126984	170.341270	519.912698	23.550458
15	1	133.752976	179.714286	85.238095	387.809524	19.732221
16	0	116.948953	214.562500	127.750000	390.208333	23.070506
17	0	116.948953	214.562500	127.750000	390.208333	23.070506
18	1	136.441799	214.041667	115.250000	579.416667	17.010413
19	0	125.074803	254.812500	147.500000	388.250000	20.224424
20	1	144.107807	186.941176	77.352941	468.411765	20.468391
21	0	121.094203	219.423077	122.423077	536.000000	30.714471
22	1	136.566667	196.388889	86.666667	364.666667	24.784379
23	1	115.891228	220.333333	103.472222	588.361111	25.091484
24	0	132.506944	206.370370	109.259259	532.962963	28.781424
25	1	121.064669	189.125000	92.575000	289.975000	14.142047

	class	GLCM_pan	Mean_Green	Mean_Red	Mean_NIR	SD_pan
26	0	138.392265	200.235294	103.294118	431.411765	22.165312
27	1	124.080000	219.159091	98.250000	638.068182	22.000223
28	1	139.506749	231.156863	105.823529	780.784314	27.496055
29	1	99.159722	241.222222	116.777778	644.222222	34.628184
...	...	...	...	...	...	...
470	1	149.716216	253.071429	135.428571	319.142857	26.842796
471	1	135.904943	222.393939	98.818182	635.151515	32.129301
472	1	126.933472	193.540984	96.229508	270.557377	17.728733
473	1	134.956962	305.000000	176.320000	289.360000	9.795182
474	1	122.770206	184.750000	90.550000	218.000000	12.226610
475	1	138.525000	208.450000	90.750000	607.150000	43.524131
476	1	123.525630	234.780822	109.397260	767.397260	16.739334
477	1	122.798595	262.777778	129.407407	737.296296	26.681991
478	1	129.784114	261.290323	161.645161	405.161290	12.148406
479	1	117.109948	1848.916667	1594.583333	1597.333333	16.612412
480	1	141.306011	190.260870	92.043478	576.304348	20.342158
481	1	139.050810	210.017699	109.150443	248.309735	14.804609
482	1	127.230618	430.281250	293.645833	341.635417	12.076363
483	1	127.029581	180.714286	79.194805	171.935065	12.660559
484	1	115.222432	225.701492	106.567164	472.641791	13.556200
485	1	131.787703	237.703704	110.333333	722.166667	23.797298
486	1	127.531250	200.250000	82.750000	374.500000	20.197463
487	1	128.181522	244.680412	140.237113	414.608247	23.704575
488	1	125.428651	279.070175	189.166667	334.254386	8.140318
489	1	125.101695	245.066667	116.933333	681.800000	17.283775

	class	GLCM_pan	Mean_Green	Mean_Red	Mean_NIR	SD_pan
490	1	129.424000	180.750000	86.000000	218.500000	12.036241
491	1	128.040752	237.850000	185.350000	275.200000	20.480723
492	1	119.971591	204.818182	98.000000	482.454545	41.877231
493	1	133.545346	234.811321	133.471698	548.716981	22.544683
494	1	129.763716	217.977012	129.793103	228.850575	19.944003
495	1	123.554348	202.826087	106.391304	364.565217	17.314068
496	1	121.549028	276.220000	175.593333	402.620000	13.394574
497	1	119.076687	247.951219	113.365854	808.024390	24.830059
498	1	107.944444	197.000000	90.000000	451.000000	8.214887
499	1	119.731928	182.238095	74.285714	301.690476	22.944278

500 rows × 6 columns

In [18]:

```
df_tes.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 [19]:

```
X_test=df_tes[['GLCM_pan', 'Mean_Green', 'Mean_Red', 'Mean_NIR', 'SD_pa
n']]
y_test=df_tes['class']
```

```
In [24]: predict=lr.predict(X_test)
```

```
In [27]: from sklearn.metrics import classification_report,confusion_matrix
```

```
In [28]: print(classification_report(y_test,predict))
```

```
             precision    recall  f1-score   support

          0       0.82      0.80      0.81       187
          1       0.88      0.89      0.89       313

avg / total       0.86      0.86      0.86       500
```

```
In [29]: from sklearn.metrics import confusion_matrix
```

```
In [30]: print(confusion_matrix(y_test,predict))
```

```
[[150  37]
 [ 34 279]]
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```