In [3]:

```python
# Importing Libraries
import pandas as pd
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
```

In [5]:

```python
# Load dataset
dataset = pd.read_csv('D:/Machine Learning Datasets/Wine.csv')
dataset
```

Out[5]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflav
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39
5	14.20	1.76	2.45	15.2	112	3.27	3.39	0.34
6	14.39	1.87	2.45	14.6	96	2.50	2.52	0.30
7	14.06	2.15	2.61	17.6	121	2.60	2.51	0.31
8	14.83	1.64	2.17	14.0	97	2.80	2.98	0.29
9	13.86	1.35	2.27	16.0	98	2.98	3.15	0.22
10	14.10	2.16	2.30	18.0	105	2.95	3.32	0.22
11	14.12	1.48	2.32	16.8	95	2.20	2.43	0.26
12	13.75	1.73	2.41	16.0	89	2.60	2.76	0.29
13	14.75	1.73	2.39	11.4	91	3.10	3.69	0.43
14	14.38	1.87	2.38	12.0	102	3.30	3.64	0.29
15	13.63	1.81	2.70	17.2	112	2.85	2.91	0.30
16	14.30	1.92	2.72	20.0	120	2.80	3.14	0.33
17	13.83	1.57	2.62	20.0	115	2.95	3.40	0.40
18	14.19	1.59	2.48	16.5	108	3.30	3.93	0.32
19	13.64	3.10	2.56	15.2	116	2.70	3.03	0.17

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflav
20	14.06	1.63	2.28	16.0	126	3.00	3.17	0.24
21	12.93	3.80	2.65	18.6	102	2.41	2.41	0.25
22	13.71	1.86	2.36	16.6	101	2.61	2.88	0.27
23	12.85	1.60	2.52	17.8	95	2.48	2.37	0.26
24	13.50	1.81	2.61	20.0	96	2.53	2.61	0.28
25	13.05	2.05	3.22	25.0	124	2.63	2.68	0.47
26	13.39	1.77	2.62	16.1	93	2.85	2.94	0.34
27	13.30	1.72	2.14	17.0	94	2.40	2.19	0.27
28	13.87	1.90	2.80	19.4	107	2.95	2.97	0.37
29	14.02	1.68	2.21	16.0	96	2.65	2.33	0.26
...	...	...	...	...	...	...	...	...
148	13.32	3.24	2.38	21.5	92	1.93	0.76	0.45
149	13.08	3.90	2.36	21.5	113	1.41	1.39	0.34
150	13.50	3.12	2.62	24.0	123	1.40	1.57	0.22
151	12.79	2.67	2.48	22.0	112	1.48	1.36	0.24
152	13.11	1.90	2.75	25.5	116	2.20	1.28	0.26
153	13.23	3.30	2.28	18.5	98	1.80	0.83	0.61
154	12.58	1.29	2.10	20.0	103	1.48	0.58	0.53
155	13.17	5.19	2.32	22.0	93	1.74	0.63	0.61
156	13.84	4.12	2.38	19.5	89	1.80	0.83	0.48
157	12.45	3.03	2.64	27.0	97	1.90	0.58	0.63
158	14.34	1.68	2.70	25.0	98	2.80	1.31	0.53
159	13.48	1.67	2.64	22.5	89	2.60	1.10	0.52
160	12.36	3.83	2.38	21.0	88	2.30	0.92	0.50
161	13.69	3.26	2.54	20.0	107	1.83	0.56	0.50
162	12.85	3.27	2.58	22.0	106	1.65	0.60	0.60
163	12.96	3.45	2.35	18.5	106	1.39	0.70	0.40
164	13.78	2.76	2.30	22.0	90	1.35	0.68	0.41
165	13.73	4.36	2.26	22.5	88	1.28	0.47	0.52
166	13.45	3.70	2.60	23.0	111	1.70	0.92	0.43
167	12.82	3.37	2.30	19.5	88	1.48	0.66	0.40
168	13.58	2.58	2.69	24.5	105	1.55	0.84	0.39
169	13.40	4.60	2.86	25.0	112	1.98	0.96	0.27
170	12.20	3.03	2.32	19.0	96	1.25	0.49	0.40
171	12.77	2.39	2.28	19.5	86	1.39	0.51	0.48

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflav
172	14.16	2.51	2.48	20.0	91	1.68	0.70	0.44
173	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52
174	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43
175	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43
176	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53
177	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56

178 rows × 14 columns

In [7]:

```
# Dataset attribures
print(dataset.shape)
print(dataset.head(5))
print(dataset.describe())
print(dataset.groupby('Customer_Segment').size())
```

```
(178, 14)
   Alcohol  Malic_Acid   Ash  Ash_Alcanity  Magnesium  Total_Phenols  \
0    14.23        1.71  2.43          15.6        127           2.80
1    13.20        1.78  2.14          11.2        100           2.65
2    13.16        2.36  2.67          18.6        101           2.80
3    14.37        1.95  2.50          16.8        113           3.85
4    13.24        2.59  2.87          21.0        118           2.80

   Flavanoids  Nonflavanoid_Phenols  Proanthocyanins  Color_Intensity   Hue
\
0        3.06                  0.28             2.29             5.64  1.04

1        2.76                  0.26             1.28             4.38  1.05

2        3.24                  0.30             2.81             5.68  1.03

3        3.49                  0.24             2.18             7.80  0.86

4        2.69                  0.39             1.82             4.32  1.04


   OD280  Proline  Customer_Segment
0   3.92     1065                 1
1   3.40     1050                 1
2   3.17     1185                 1
3   3.45     1480                 1
4   2.93      735                 1
          Alcohol  Malic_Acid         Ash  Ash_Alcanity   Magnesium  \
count  178.000000  178.000000  178.000000    178.000000  178.000000
mean    13.000618    2.336348    2.366517     19.494944   99.741573
std      0.811827    1.117146    0.274344      3.339564   14.282484
min     11.030000    0.740000    1.360000     10.600000   70.000000
25%     12.362500    1.602500    2.210000     17.200000   88.000000
50%     13.050000    1.865000    2.360000     19.500000   98.000000
75%     13.677500    3.082500    2.557500     21.500000  107.000000
max     14.830000    5.800000    3.230000     30.000000  162.000000

       Total_Phenols  Flavanoids  Nonflavanoid_Phenols  Proanthocyanins  \
count     178.000000  178.000000            178.000000       178.000000
```

```
mean        2.295112     2.029270            0.361854          1.590899
std         0.625851     0.998859            0.124453          0.572359
min         0.980000     0.340000            0.130000          0.410000
25%         1.742500     1.205000            0.270000          1.250000
50%         2.355000     2.135000            0.340000          1.555000
75%         2.800000     2.875000            0.437500          1.950000
max         3.880000     5.080000            0.660000          3.580000

       Color_Intensity          Hue        OD280       Proline
Customer_Segment
count       178.000000   178.000000   178.000000    178.000000         178.0000(
0
mean          5.058090     0.957449     2.611685    746.893258           1.9382(
2
std           2.318286     0.228572     0.709990    314.907474           0.7750:
5
min           1.280000     0.480000     1.270000    278.000000           1.0000(
0
25%           3.220000     0.782500     1.937500    500.500000           1.0000(
0
50%           4.690000     0.965000     2.780000    673.500000           2.0000(
0
75%           6.200000     1.120000     3.170000    985.000000           3.0000(
0
max          13.000000     1.710000     4.000000   1680.000000           3.0000(
0
Customer_Segment
1    59
2    71
3    48
dtype: int64
```

In [9]:

```
# Splitting the dataset
array= dataset.values
array
```

Out[9]:

```
array([[  1.42300000e+01,   1.71000000e+00,   2.43000000e+00, ...,
          3.92000000e+00,   1.06500000e+03,   1.00000000e+00],
       [  1.32000000e+01,   1.78000000e+00,   2.14000000e+00, ...,
          3.40000000e+00,   1.05000000e+03,   1.00000000e+00],
       [  1.31600000e+01,   2.36000000e+00,   2.67000000e+00, ...,
          3.17000000e+00,   1.18500000e+03,   1.00000000e+00],
       ...,
       [  1.32700000e+01,   4.28000000e+00,   2.26000000e+00, ...,
          1.56000000e+00,   8.35000000e+02,   3.00000000e+00],
       [  1.31700000e+01,   2.59000000e+00,   2.37000000e+00, ...,
          1.62000000e+00,   8.40000000e+02,   3.00000000e+00],
       [  1.41300000e+01,   4.10000000e+00,   2.74000000e+00, ...,
          1.60000000e+00,   5.60000000e+02,   3.00000000e+00]])
```

In [22]:

```
x = array[:,0:13]
Y = array[:,13]
seed=11
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(x, Y,
```

```
test_size=0.30, random_state=seed)
```

In [25]:

```
#Building models
seed = 11
scoring = 'accuracy'
models = []
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('NB', GaussianNB()))


# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X_train, Y_train, c
v=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
LDA: 0.983974 (0.032083)
NB: 0.975641 (0.037246)
```

In [26]:

```
# Make predictions on test dataset
NB = GaussianNB()
NB.fit(X_train, Y_train)
predictions = NB.predict(X_test)
print("Naive Baiyes" , accuracy_score(Y_test, predictions))
print(confusion_matrix(Y_test, predictions))
print(classification_report(Y_test, predictions))
```

```
Naive Baiyes 0.944444444444
[[21  0  0]
 [ 1 19  2]
 [ 0  0 11]]
             precision    recall  f1-score   support

        1.0       0.95      1.00      0.98        21
        2.0       1.00      0.86      0.93        22
        3.0       0.85      1.00      0.92        11

avg / total       0.95      0.94      0.94        54
```

In [28]:

```
#LDA
LDA = LinearDiscriminantAnalysis()
LDA.fit(X_train, Y_train)
predictions1 = LDA.predict(X_test)
print("LDA", accuracy_score(Y_test, predictions1))
print(confusion_matrix(Y_test, predictions1))
print(classification_report(Y_test, predictions1))
```

```
LDA 1.0
```

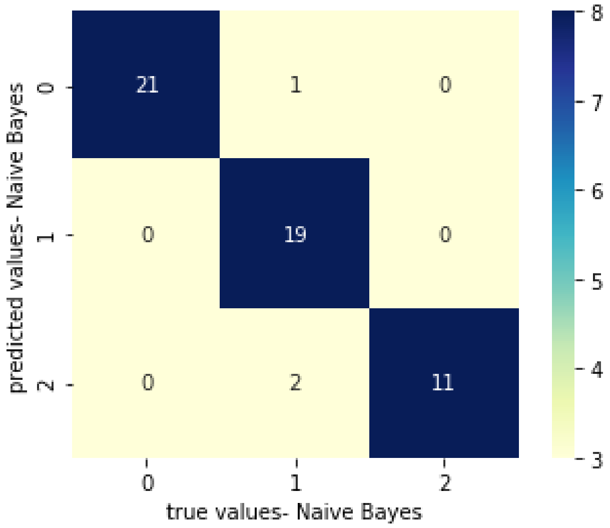
```
[[21  0  0]
 [ 0 22  0]
 [ 0  0 11]]
             precision    recall  f1-score   support

        1.0       1.00      1.00      1.00        21
        2.0       1.00      1.00      1.00        22
        3.0       1.00      1.00      1.00        11

avg / total       1.00      1.00      1.00        54
```

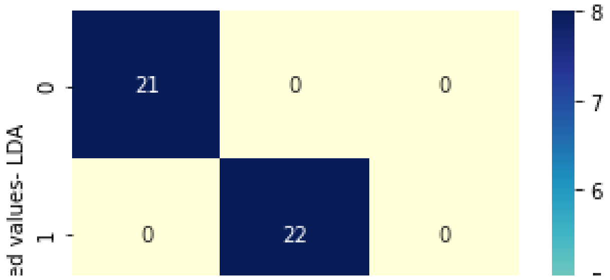
In [30]:

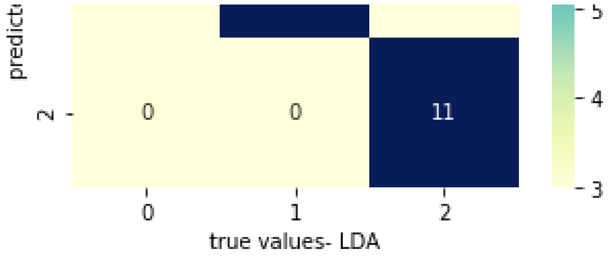
```python
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
mat = confusion_matrix(Y_test, predictions)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=True,
cmap="YlGnBu",vmin=3,vmax=8)
plt.xlabel('true values- Naive Bayes')
plt.ylabel('predicted values- Naive Bayes')
plt.show()
```



In [31]:

```python
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
mat = confusion_matrix(Y_test, predictions1)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=True,
cmap="YlGnBu",vmin=3,vmax=8)
plt.xlabel('true values- LDA')
plt.ylabel('predicted values- LDA')
plt.show()
```





In [ ]:

```
#The model accuracy of LDA is greater as compared to model accuracy of Naiv
e Bayes
```