In [1]:
```
# Importing the required libraries for visualization
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import os
os.chdir('data')

# Visualization Prefrences.
%matplotlib inline
sns.set_style("whitegrid")
plt.style.use("fivethirtyeight")
```

In [2]:
```
%pip install xlrd==1.2.0
```

```
Requirement already satisfied: xlrd==1.2.0 in c:\users\mohamad\appdata\local\prog
rams\python\python36\lib\site-packages (1.2.0)
Note: you may need to restart the kernel to use updated packages.
```

01- Data Description Section

In [3]:
```
# Data Retrieving
excel_file_path = "DryBeanDataset\Dry_Bean_Dataset.xlsx"

df = pd.read_excel(excel_file_path,  sheet_name="sheet1", )
df.head(8).T
```

Out[3]:

	0	1	2	3	4	5	
Area	28395	28734	29380	30008	30140	30279	3
Perimeter	610.291	638.018	624.11	645.884	620.134	634.927	67
MajorAxisLength	208.178	200.525	212.826	210.558	201.848	212.561	2
MinorAxisLength	173.889	182.734	175.931	182.517	190.279	181.51	18
AspectRation	1.19719	1.09736	1.20971	1.15364	1.0608	1.17107	1.1
Eccentricity	0.549812	0.411785	0.562727	0.498616	0.33368	0.520401	0.48
ConvexArea	28715	29172	29690	30724	30417	30600	3
EquivDiameter	190.141	191.273	193.411	195.467	195.897	196.348	19
Extent	0.763923	0.783968	0.778113	0.782681	0.773098	0.775688	0.76
Solidity	0.988856	0.984986	0.989559	0.976696	0.990893	0.98951	0.98
roundness	0.958027	0.887034	0.947849	0.903936	0.984877	0.943852	0.8
Compactness	0.913358	0.953861	0.908774	0.928329	0.970516	0.923726	0.93
ShapeFactor1	0.00733151	0.00697866	0.00724391	0.00701673	0.00669701	0.00702007	0.006
ShapeFactor2	0.00314729	0.00356362	0.00304773	0.00321456	0.00366497	0.00315278	0.0032
ShapeFactor3	0.834222	0.909851	0.825871	0.861794	0.9419	0.85327	0.87
ShapeFactor4	0.998724	0.99843	0.999066	0.994199	0.999166	0.999236	0.99
Class	SEKER	SEKER	SEKER	SEKER	SEKER	SEKER	S

In [4]:
```
df.shape
```

Out[4]:
```
(13611, 17)
```

In [5]:
```
# Extract Descriptive Data.
pd.set_option("display.float", "{:.2f}".format)
df.describe().T
```

Out[5]:

	count	mean	std	min	25%	50%	75%	max
Area	13611.00	53048.28	29324.10	20420.00	36328.00	44652.00	61332.00	254616.00
Perimeter	13611.00	855.28	214.29	524.74	703.52	794.94	977.21	1985.37
MajorAxisLength	13611.00	320.14	85.69	183.60	253.30	296.88	376.50	738.86
MinorAxisLength	13611.00	202.27	44.97	122.51	175.85	192.43	217.03	460.20
AspectRation	13611.00	1.58	0.25	1.02	1.43	1.55	1.71	2.43
Eccentricity	13611.00	0.75	0.09	0.22	0.72	0.76	0.81	0.91
ConvexArea	13611.00	53768.20	29774.92	20684.00	36714.50	45178.00	62294.00	263261.00
EquivDiameter	13611.00	253.06	59.18	161.24	215.07	238.44	279.45	569.37
Extent	13611.00	0.75	0.05	0.56	0.72	0.76	0.79	0.87
Solidity	13611.00	0.99	0.00	0.92	0.99	0.99	0.99	0.99
roundness	13611.00	0.87	0.06	0.49	0.83	0.88	0.92	0.99
Compactness	13611.00	0.80	0.06	0.64	0.76	0.80	0.83	0.99
ShapeFactor1	13611.00	0.01	0.00	0.00	0.01	0.01	0.01	0.01
ShapeFactor2	13611.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ShapeFactor3	13611.00	0.64	0.10	0.41	0.58	0.64	0.70	0.97
ShapeFactor4	13611.00	1.00	0.00	0.95	0.99	1.00	1.00	1.00

02- Exploratory Data Analysis (EDA) & Feature Engineering Section

In [6]:
```
# Check for Null Values
df.isna().sum()
```

Out[6]:
```
Area               0
Perimeter          0
MajorAxisLength    0
MinorAxisLength    0
AspectRation       0
Eccentricity       0
ConvexArea         0
EquivDiameter      0
Extent             0
Solidity           0
```

```
roundness          0
Compactness        0
ShapeFactor1       0
ShapeFactor2       0
ShapeFactor3       0
ShapeFactor4       0
Class              0
dtype: int64
```

In [7]:

```
df.dtypes
```

Out[7]:

```
Area                int64
Perimeter         float64
MajorAxisLength   float64
MinorAxisLength   float64
AspectRation      float64
Eccentricity      float64
ConvexArea          int64
EquivDiameter     float64
Extent            float64
Solidity          float64
roundness         float64
Compactness       float64
ShapeFactor1      float64
ShapeFactor2      float64
ShapeFactor3      float64
ShapeFactor4      float64
Class              object
dtype: object
```

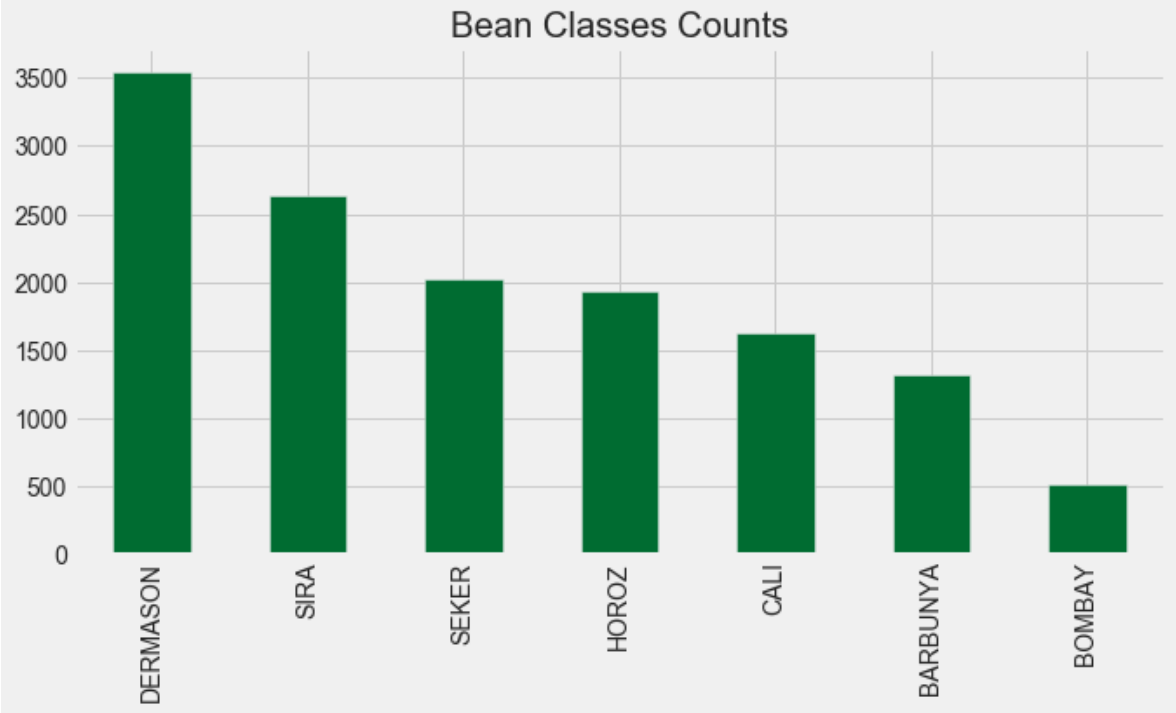
In [8]:

```
df_class_counts = df.Class.value_counts().to_frame()
print(df_class_counts)
df.Class.value_counts().plot(kind="bar", color='#006C31', figsize=(10,5), title =
```

```
          Class
DERMASON   3546
SIRA       2636
SEKER      2027
HOROZ      1928
CALI       1630
BARBUNYA   1322
BOMBAY      522
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2df0111b320>
```



In [9]:

```
# Check the percentage of each class
df_class_perc = df.Class.value_counts(normalize=True).to_frame()
df_class_perc["Class"] = df_class_perc["Class"] * 100
df_class_perc.rename(columns = {"Class": "Class%"}, inplace=True)
df_class_perc
```

Out[9]:

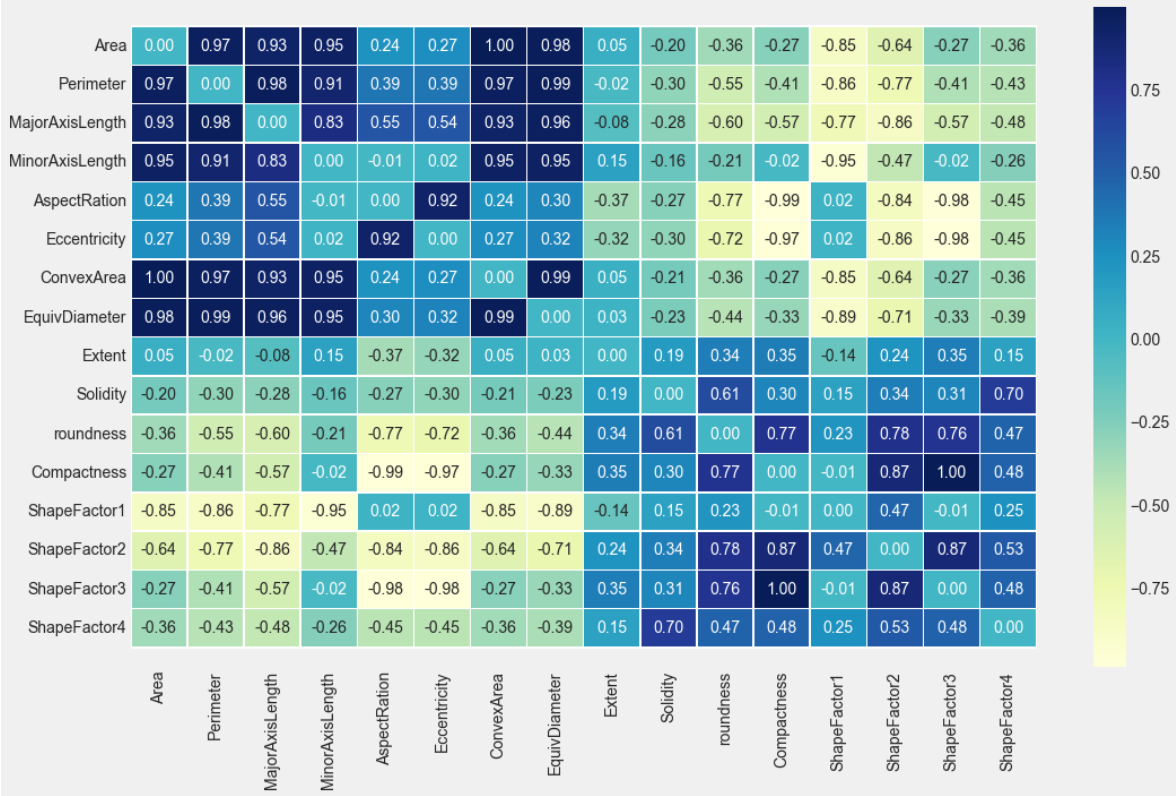
	Class%
DERMASON	26.05
SIRA	19.37
SEKER	14.89
HOROZ	14.17
CALI	11.98
BARBUNYA	9.71
BOMBAY	3.84

In [10]:

```
# Studying the correlations between features using Heat Map!
corr_matrix = df.corr()
for x in range(corr_matrix.shape[0]):
    corr_matrix.iloc[x,x] = 0.0

fig, ax = plt.subplots(figsize=(16, 10))
ax = sns.heatmap(corr_matrix,
                 annot=True,
                 linewidths=0.5,
                 fmt=".2f",
                 cmap="YlGnBu");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

Out[10]:

```
(16.5, -0.5)
```



In [11]:

```
# The correlation matrix
corr_mat = df.corr()

# Strip out the diagonal values for the next step
for x in range(corr_mat.shape[0]):
    corr_mat.iloc[x,x] = 0.0

corr_mat
```

Out[11]:

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricit
Area	0.00	0.97	0.93	0.95	0.24	0.2
Perimeter	0.97	0.00	0.98	0.91	0.39	0.3
MajorAxisLength	0.93	0.98	0.00	0.83	0.55	0.5
MinorAxisLength	0.95	0.91	0.83	0.00	-0.01	0.0
AspectRation	0.24	0.39	0.55	-0.01	0.00	0.9
Eccentricity	0.27	0.39	0.54	0.02	0.92	0.0
ConvexArea	1.00	0.97	0.93	0.95	0.24	0.2
EquivDiameter	0.98	0.99	0.96	0.95	0.30	0.3
Extent	0.05	-0.02	-0.08	0.15	-0.37	-0.3
Solidity	-0.20	-0.30	-0.28	-0.16	-0.27	-0.3
roundness	-0.36	-0.55	-0.60	-0.21	-0.77	-0.7
Compactness	-0.27	-0.41	-0.57	-0.02	-0.99	-0.9
ShapeFactor1	-0.85	-0.86	-0.77	-0.95	0.02	0.0
ShapeFactor2	-0.64	-0.77	-0.86	-0.47	-0.84	-0.8
ShapeFactor3	-0.27	-0.41	-0.57	-0.02	-0.98	-0.9

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricit
ShapeFactor4	-0.36	-0.43	-0.48	-0.26	-0.45	-0.4

In [12]:

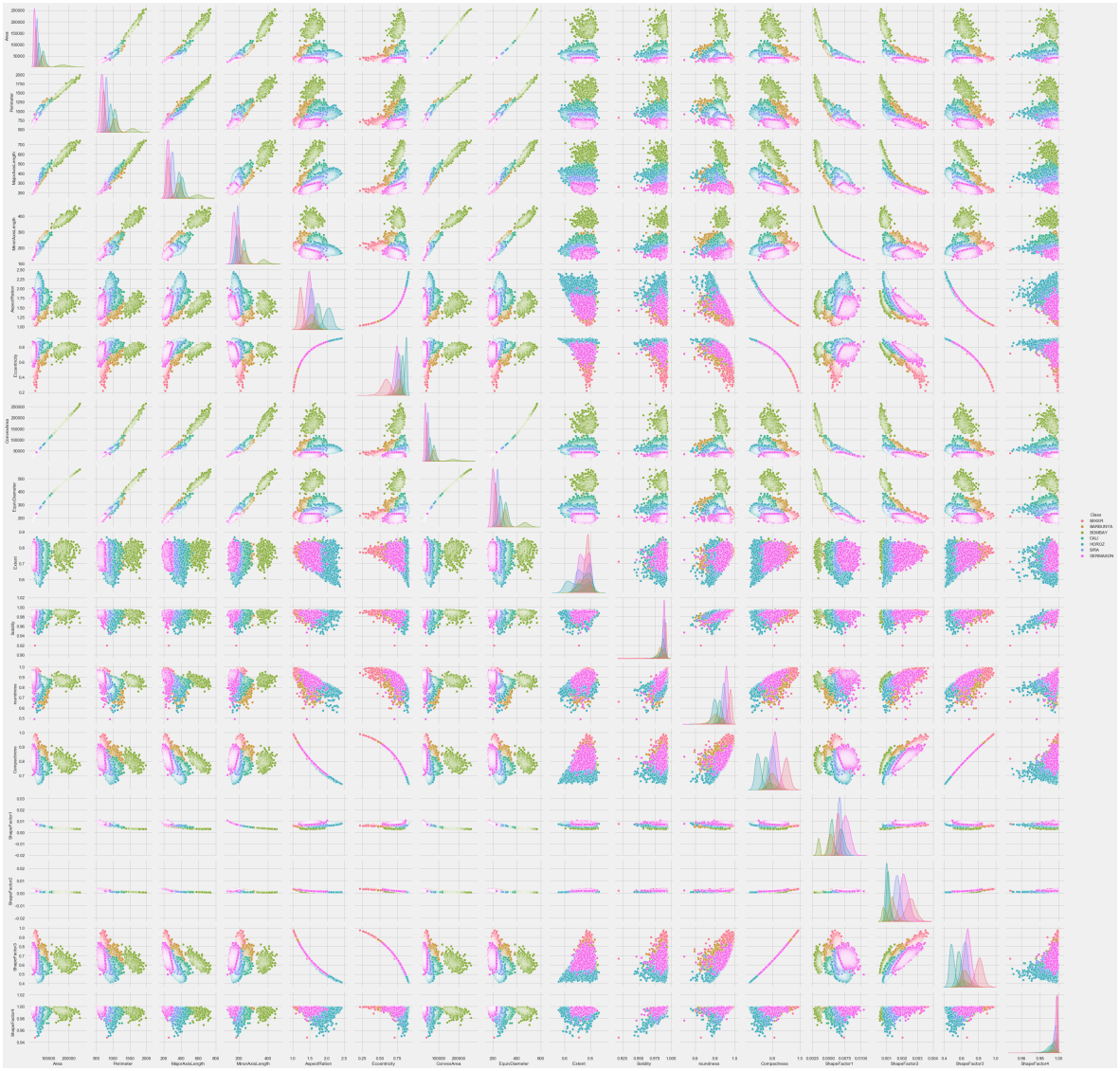
```
# Pairwise maximal correlations
corr_max  = corr_mat.abs().max().to_frame()
corr_id_max = corr_mat.abs().idxmax().to_frame()

# dataframe aggrigation and processing
pair_features_corr = pd.merge(corr_id_max, corr_max, on = corr_max.index)
pair_features_corr = pair_features_corr.rename(columns = {'key_0':'Feature_one',
                                                          .sort_values('correlation', ascer
                                                          .reset_index().drop('index', axis
pair_features_corr
```

Out[12]:

	Feature_one	Feature_two	correlation
0	Area	ConvexArea	1.00
1	ConvexArea	Area	1.00
2	Compactness	ShapeFactor3	1.00
3	ShapeFactor3	Compactness	1.00
4	Perimeter	EquivDiameter	0.99
5	EquivDiameter	Perimeter	0.99
6	AspectRation	Compactness	0.99
7	Eccentricity	ShapeFactor3	0.98
8	MajorAxisLength	Perimeter	0.98
9	MinorAxisLength	Area	0.95
10	ShapeFactor1	MinorAxisLength	0.95
11	ShapeFactor2	ShapeFactor3	0.87
12	roundness	ShapeFactor2	0.78
13	Solidity	ShapeFactor4	0.70
14	ShapeFactor4	Solidity	0.70
15	Extent	AspectRation	0.37

In [13]:

```
float_columns = [col for col in df.columns if col != 'Class']

sns.set_context('notebook')
sns.pairplot(df[float_columns + ['Class']],
             hue='Class'
             );
### END SOLUTION
```



And an examination of the skew values in anticipation of transformations.

- 0 : no skew
- pos : right skew
- neg : left skew

In [14]:
```
skew_columns = (df
                .skew()
                .sort_values(ascending=False)).to_frame("skewness_value")
skew_columns
```

Out[14]:

	skewness_value
Area	2.95
ConvexArea	2.94
MinorAxisLength	2.24
EquivDiameter	1.95
Perimeter	1.63
MajorAxisLength	1.36
AspectRation	0.58

	skewness_value
ShapeFactor2	0.30
ShapeFactor3	0.24
Compactness	0.04
ShapeFactor1	-0.53
roundness	-0.64
Extent	-0.90
Eccentricity	-1.06
Solidity	-2.55
ShapeFactor4	-2.76

In [15]:
```
skew_columns = (df
                .skew()
                .sort_values(ascending=False)).to_frame("skewness_value")
skew_columns = skew_columns.query('skewness_value > 0.75')
skew_columns
```

Out[15]:

	skewness_value
Area	2.95
ConvexArea	2.94
MinorAxisLength	2.24
EquivDiameter	1.95
Perimeter	1.63
MajorAxisLength	1.36

In [16]:
```
# Perform log transform on skewed columns
for col in skew_columns['skewness_value'].index.tolist():
    df[col] = np.log1p(df[col])
```

In [17]:
```
skew_trans_columns = (df
                .skew()
                .sort_values(ascending=False)).to_frame("skewness_value")
skew_trans_columns
```

Out[17]:

	skewness_value
MinorAxisLength	1.31
EquivDiameter	1.07
Area	1.07
ConvexArea	1.07
Perimeter	0.84
MajorAxisLength	0.63
AspectRation	0.58

	skewness_value
ShapeFactor2	0.30
ShapeFactor3	0.24
Compactness	0.04
ShapeFactor1	-0.53
roundness	-0.64
Extent	-0.90
Eccentricity	-1.06
Solidity	-2.55
ShapeFactor4	-2.76

In [18]:
```
df.dtypes
```

Out[18]:
```
Area                float64
Perimeter           float64
MajorAxisLength     float64
MinorAxisLength     float64
AspectRation        float64
Eccentricity        float64
ConvexArea          float64
EquivDiameter       float64
Extent              float64
Solidity            float64
roundness           float64
Compactness         float64
ShapeFactor1        float64
ShapeFactor2        float64
ShapeFactor3        float64
ShapeFactor4        float64
Class                object
dtype: object
```

In [19]:
```
df.head(7)
```

Out[19]:

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	ConvexArea
0	10.25	6.42	5.34	5.16	1.20	0.55	10.27
1	10.27	6.46	5.31	5.21	1.10	0.41	10.28
2	10.29	6.44	5.37	5.18	1.21	0.56	10.30
3	10.31	6.47	5.35	5.21	1.15	0.50	10.33
4	10.31	6.43	5.31	5.25	1.06	0.33	10.32
5	10.32	6.46	5.36	5.21	1.17	0.52	10.33
6	10.32	6.51	5.36	5.22	1.15	0.49	10.34

In [20]:
```
from sklearn.preprocessing import StandardScaler
float_columns = [col for col in df.columns if col != 'Class']
sc = StandardScaler()
```

```
df[float_columns] = sc.fit_transform(df[float_columns])
df.head(7)
```

Out[20]:

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	ConvexArea
0	-1.28	-1.38	-1.63	-0.68	-1.57	-2.19	-1.28
1	-1.25	-1.19	-1.78	-0.43	-1.97	-3.69	-1.24
2	-1.20	-1.28	-1.54	-0.62	-1.51	-2.05	-1.20
3	-1.15	-1.13	-1.58	-0.43	-1.74	-2.74	-1.12
4	-1.14	-1.31	-1.75	-0.21	-2.12	-4.54	-1.14
5	-1.12	-1.21	-1.54	-0.46	-1.67	-2.51	-1.13
6	-1.11	-0.97	-1.57	-0.39	-1.77	-2.84	-1.10

03- Machine Learning Section : clusterning methods

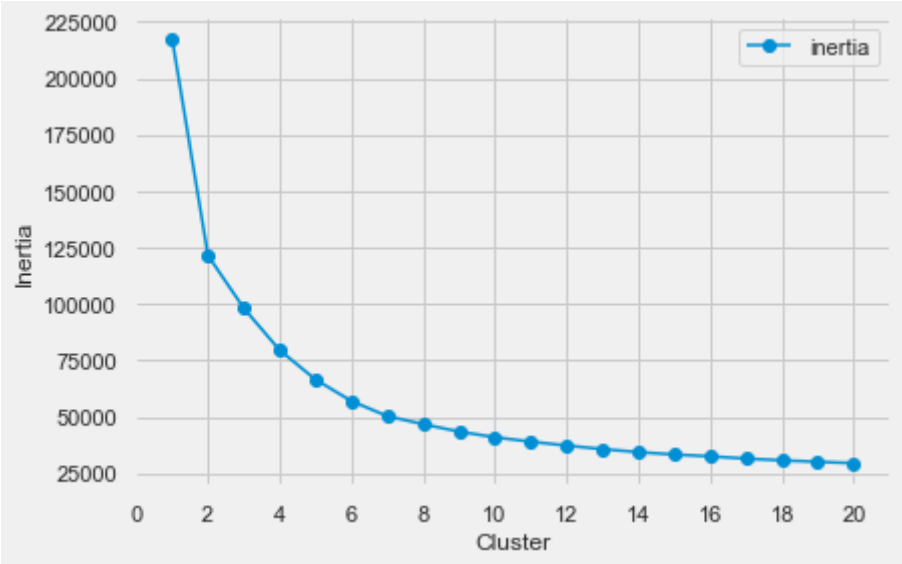
1- K-means Algorithm

In [21]:

```
### BEGIN SOLUTION
from sklearn.cluster import KMeans

# Create and fit a range of models
km_list = list()

for clust in range(1,21):
    km = KMeans(n_clusters=clust, random_state=42)
    km = km.fit(df[float_columns])

    km_list.append(pd.Series({'clusters': clust,
                              'inertia': km.inertia_,
                              'model': km}))
```

In [22]:

```
plot_data = (pd.concat(km_list, axis=1)
             .T
             [['clusters','inertia']]
             .set_index('clusters'))

ax = plot_data.plot(marker='o',ls='-')
ax.set_xticks(range(0,21,2))
ax.set_xlim(0,21)
ax.set(xlabel='Cluster', ylabel='Inertia');
### END SOLUTION
```



In [23]:
```
Bean_classes = pd.merge(df_class_counts, df_class_perc, on = df_class_counts.inde
Bean_classes.rename(columns={'key_0':'class_name'}, inplace=True)
Bean_classes
```

Out[23]:

	class_name	Class	Class%
0	DERMASON	3546	26.05
1	SIRA	2636	19.37
2	SEKER	2027	14.89
3	HOROZ	1928	14.17
4	CALI	1630	11.98
5	BARBUNYA	1322	9.71
6	BOMBAY	522	3.84

In [24]:
```
### BEGIN SOLUTION
km = KMeans(n_clusters=8, random_state=42)
km = km.fit(df[float_columns])
df['k-means'] = km.predict(df[float_columns])
df.sample(7)
```

Out[24]:

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	Convex
13245	-0.53	-0.55	-0.54	-0.46	-0.35	-0.08	
5554	-0.52	-0.23	0.12	-1.22	1.84	1.31	
8729	-0.19	-0.24	-0.34	0.05	-0.63	-0.41	
8712	-0.19	-0.23	-0.14	-0.23	-0.00	0.26	
4990	1.22	1.08	1.11	1.21	0.18	0.41	
8360	-0.28	-0.32	-0.27	-0.26	-0.17	0.10	
9099	-0.10	-0.12	-0.02	-0.20	0.14	0.38	

In [25]:

```
# Group by Class and K-means for comparsion between clustered classes and actual
(df[['Class','k-means']]
 .groupby(['Class','k-means'])
 .size()
 .to_frame()
 .rename(columns={0:'number'}))
### END SOLUTION
```

Out[25]:

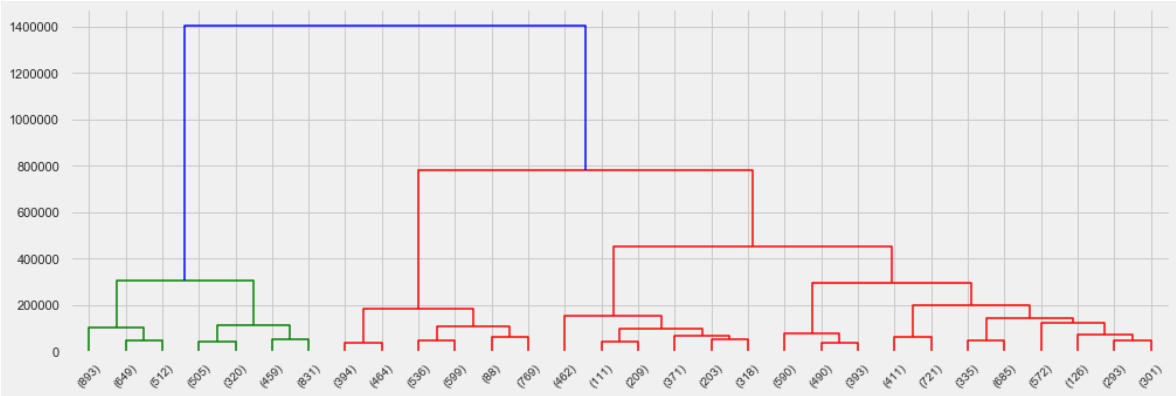
Class	k-means	number
BARBUNYA	0	6
	1	8
	2	1006
	5	61
	6	216
	7	25
BOMBAY	4	520
	7	2
CALI	0	2
	1	11
	2	24
	4	1
	5	14
	6	1364
	7	214
DERMASON	0	114
	1	5
	3	2726
	5	694
	7	7
HOROZ	1	1630
	3	3
	5	47
	6	68
	7	180
SEKER	0	1875
	2	12
	3	13
	5	125
	7	2

		number
Class	k-means	
	0	22
	1	78
	2	6
SIRA	3	68
	5	2423
	6	17
	7	22

2- Agglomerative Algorithm

In [26]:
```
from sklearn.cluster import AgglomerativeClustering
### BEGIN SOLUTION
ag = AgglomerativeClustering(n_clusters=7, linkage='ward', compute_full_tree=True
ag = ag.fit(df[float_columns])
df['agglom'] = ag.fit_predict(df[float_columns])
```

In [27]:
```
# First, we import the cluster hierarchy module from SciPy (described above) to c
from scipy.cluster import hierarchy

Z = hierarchy.linkage(ag.children_, method='ward')

fig, ax = plt.subplots(figsize=(15,5))

den = hierarchy.dendrogram(Z, orientation='top',
                           p=30, truncate_mode='lastp',
                           show_leaf_counts=True, ax=ax,
                           above_threshold_color='blue')
### END SOLUTION
```



In [28]:
```
# First, for Agglomerative Clustering:
(df[['Class','agglom']]
 .groupby(['Class','agglom'])
 .size()
 .to_frame()
 .rename(columns={0:'number'}))
```

Out[28]:

		number

Class	agglom	number
Class	**agglom**	
BARBUNYA	**0**	1273
	1	2
	2	32
	3	6
	5	8
	6	1
BOMBAY	**4**	522
CALI	**0**	1572
	1	32
	2	18
	3	2
	6	6
DERMASON	**0**	1
	1	1
	2	494
	3	80
	5	2969
	6	1
HOROZ	**0**	52
	1	1602
	2	167
	5	12
	6	95
SEKER	**0**	7
	2	60
	3	1879
	5	80
	6	1
SIRA	**0**	22
	1	33
	2	2252
	3	57
	5	266
	6	6

3- MeanShift Algorithm

In [462...
```
from sklearn.cluster import MeanShift
ms  = MeanShift(bandwidth=2.8, n_jobs=-1)
ms = ms.fit(df[float_columns])
```

In [463...
```
np.unique(ms.labels_)
```

Out[463...
```
array([0, 1, 2, 3, 4], dtype=int64)
```

In [464...
```
df['MeanShift'] = ms.fit_predict(df[float_columns])
```

In [465...
```
(df[['Class','MeanShift']]
 .groupby(['Class','MeanShift'])
 .size()
 .to_frame()
 .rename(columns={0:'number'}))
```

Out[465...

		number
Class	**MeanShift**	
BARBUNYA	**0**	974
	1	346
	2	1
	4	1
BOMBAY	**1**	508
	2	14
CALI	**0**	891
	1	710
	2	29
DERMASON	**0**	3539
	4	7
HOROZ	**0**	1864
	1	3
	2	22
	3	4
	4	35
SEKER	**0**	2026
	3	1
SIRA	**0**	2636

4- DBSCAN Algorithm

In [500…]
```
from sklearn.cluster import DBSCAN
dbs = DBSCAN(eps=0.5, min_samples=11, metric='euclidean')
dbs = dbs.fit(df[float_columns])
```

In [501…]
```
np.unique(dbs.labels_)
```

Out[501…]
```
array([-1,  0,  1,  2,  3,  4,  5,  6], dtype=int64)
```

In [502…]
```
df['dbscan'] = dbs.fit_predict(df[float_columns])
```

In [503…]
```
# First, for Agglomerative Clustering:
(df[['Class','dbscan']]
 .groupby(['Class','dbscan'])
 .size()
 .to_frame()
 .rename(columns={0:'number'}))
```

Out[503…]

Class	dbscan	number
BARBUNYA	-1	1319
	0	3
BOMBAY	-1	522
CALI	-1	1589
	0	4
	1	37
DERMASON	-1	1264
	0	2277
	6	5
HOROZ	-1	1869
	0	8
	2	11
	3	30
	4	8
	5	1
	6	1
SEKER	-1	410
	0	1617
SIRA	-1	1048
	0	1578
	5	9
	6	1

In [ ]:

In [ ]: