Import libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import (StandardScaler,LabelEncoder)
```

Load the data

In [26]:
```
df= pd.read_csv("D:\\data.csv")
df.head()
```

Out[26]:

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	ConvexArea	EquivDiame
0	28395	610.291	208.178117	173.888747	1.197191	0.549812	28715	190.141
1	28734	638.018	200.524796	182.734419	1.097356	0.411785	29172	191.272
2	29380	624.110	212.826130	175.931143	1.209713	0.562727	29690	193.410
3	30008	645.884	210.557999	182.516516	1.153638	0.498616	30724	195.467
4	30140	620.134	201.847882	190.279279	1.060798	0.333680	30417	195.896

Shape, Size , Info, Describe of data

In [5]:
```
df.shape
```

Out[5]: (13611, 17)

In [7]:
```
df.size
```

Out[7]: 231387

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13611 entries, 0 to 13610
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Area             13611 non-null  int64  
 1   Perimeter        13611 non-null  float64
 2   MajorAxisLength  13611 non-null  float64
 3   MinorAxisLength  13611 non-null  float64
 4   AspectRation     13611 non-null  float64
 5   Eccentricity     13611 non-null  float64
 6   ConvexArea       13611 non-null  int64  
 7   EquivDiameter    13611 non-null  float64
 8   Extent           13611 non-null  float64
 9   Solidity         13611 non-null  float64
 10  roundness        13611 non-null  float64
 11  Compactness      13611 non-null  float64
 12  ShapeFactor1     13611 non-null  float64
 13  ShapeFactor2     13611 non-null  float64
 14  ShapeFactor3     13611 non-null  float64
 15  ShapeFactor4     13611 non-null  float64
 16  Class            13611 non-null  object 
dtypes: float64(14), int64(2), object(1)
memory usage: 1.8+ MB
```

Summary

1- There are 13611 records with 17 variables

2- The Dataset have not any null values and null values

In [9]:
```
df.describe()
```

Out[9]:

Eccentricity	ConvexArea	EquivDiameter	Extent	Solidity	roundness	Compactness	Shapel
13611.000000	13611.000000	13611.000000	13611.000000	13611.000000	13611.000000	13611.000000	13611.
0.750895	53768.200206	253.064220	0.749733	0.987143	0.873282	0.799864	0.
0.092002	29774.915817	59.177120	0.049086	0.004660	0.059520	0.061713	0
0.218951	20684.000000	161.243764	0.555315	0.919246	0.489618	0.640577	0.
0.715928	36714.500000	215.068003	0.718634	0.985670	0.832096	0.762469	0.
0.764441	45178.000000	238.438026	0.759859	0.988283	0.883157	0.801277	0.
0.810466	62294.000000	279.446467	0.786851	0.990013	0.916869	0.834270	0.
0.911423	263261.000000	569.374358	0.866195	0.994677	0.990685	0.987303	0.

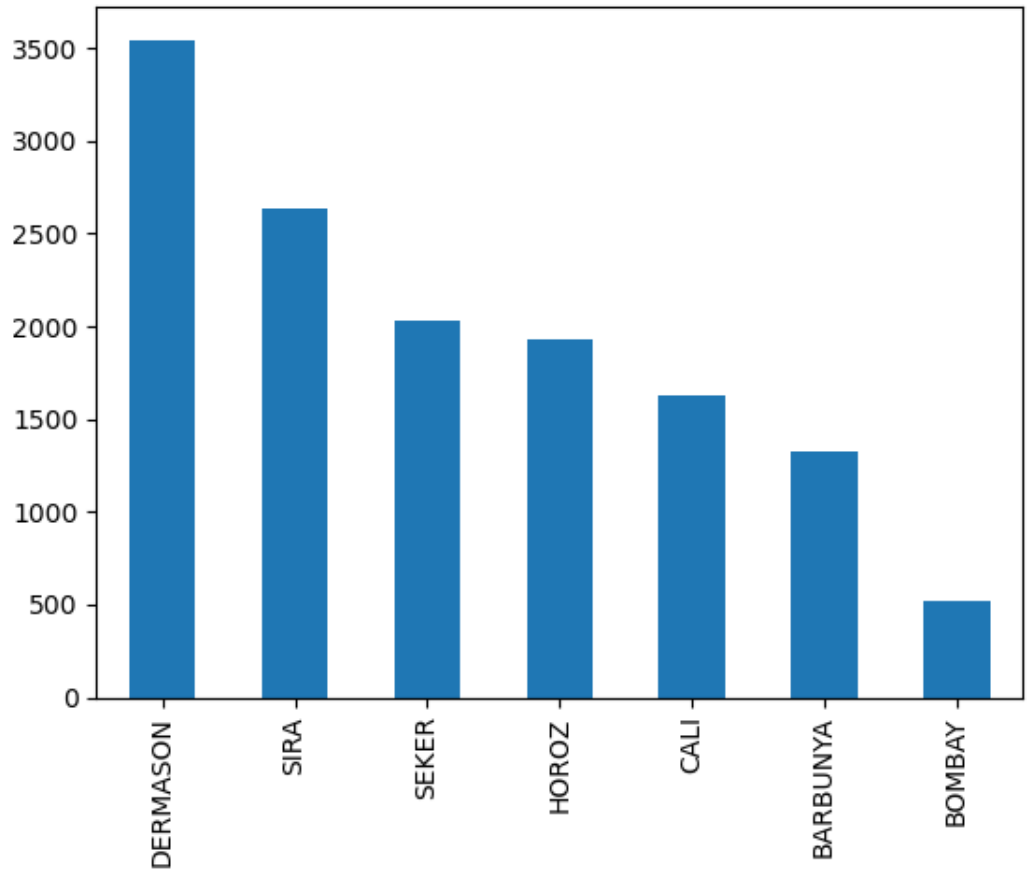
Counts the value in class columns

In [10]:
```
df.Class.value_counts()
```

Out[10]:
```
DERMASON    3546
SIRA        2636
SEKER       2027
HOROZ       1928
CALI        1630
BARBUNYA    1322
BOMBAY       522
Name: Class, dtype: int64
```

In [13]:
```
df['Class'].value_counts().plot.bar()
```

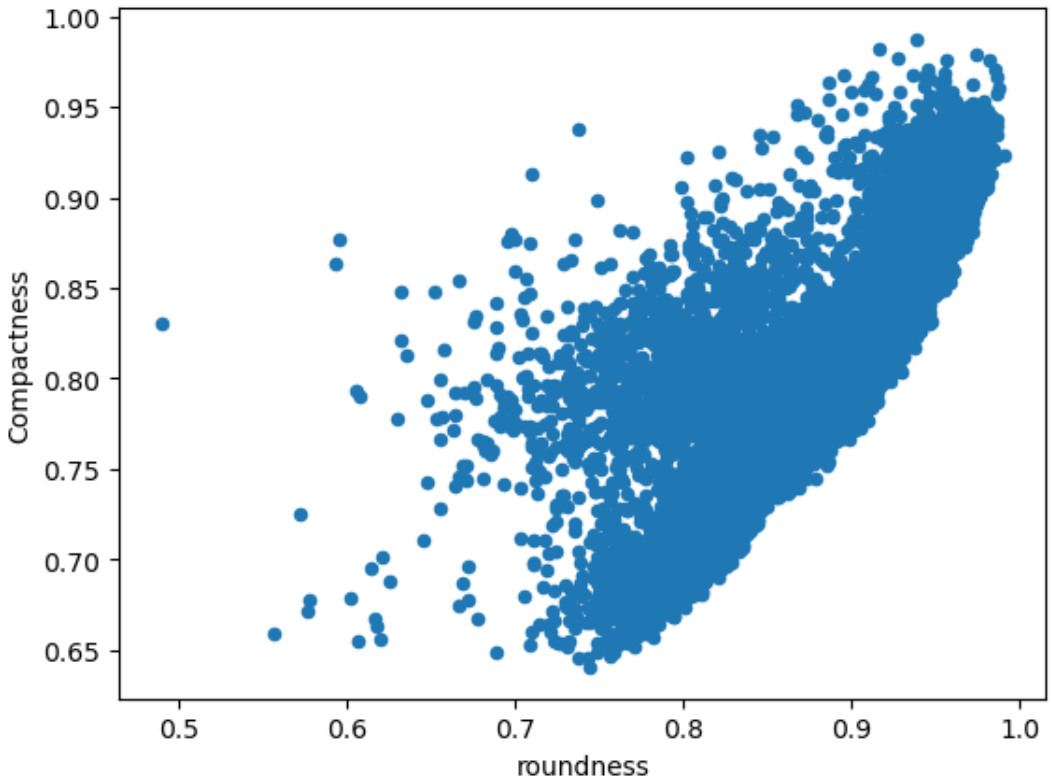
Out[13]:
```
<AxesSubplot:>
```



Visualization between Roundness and Compactness

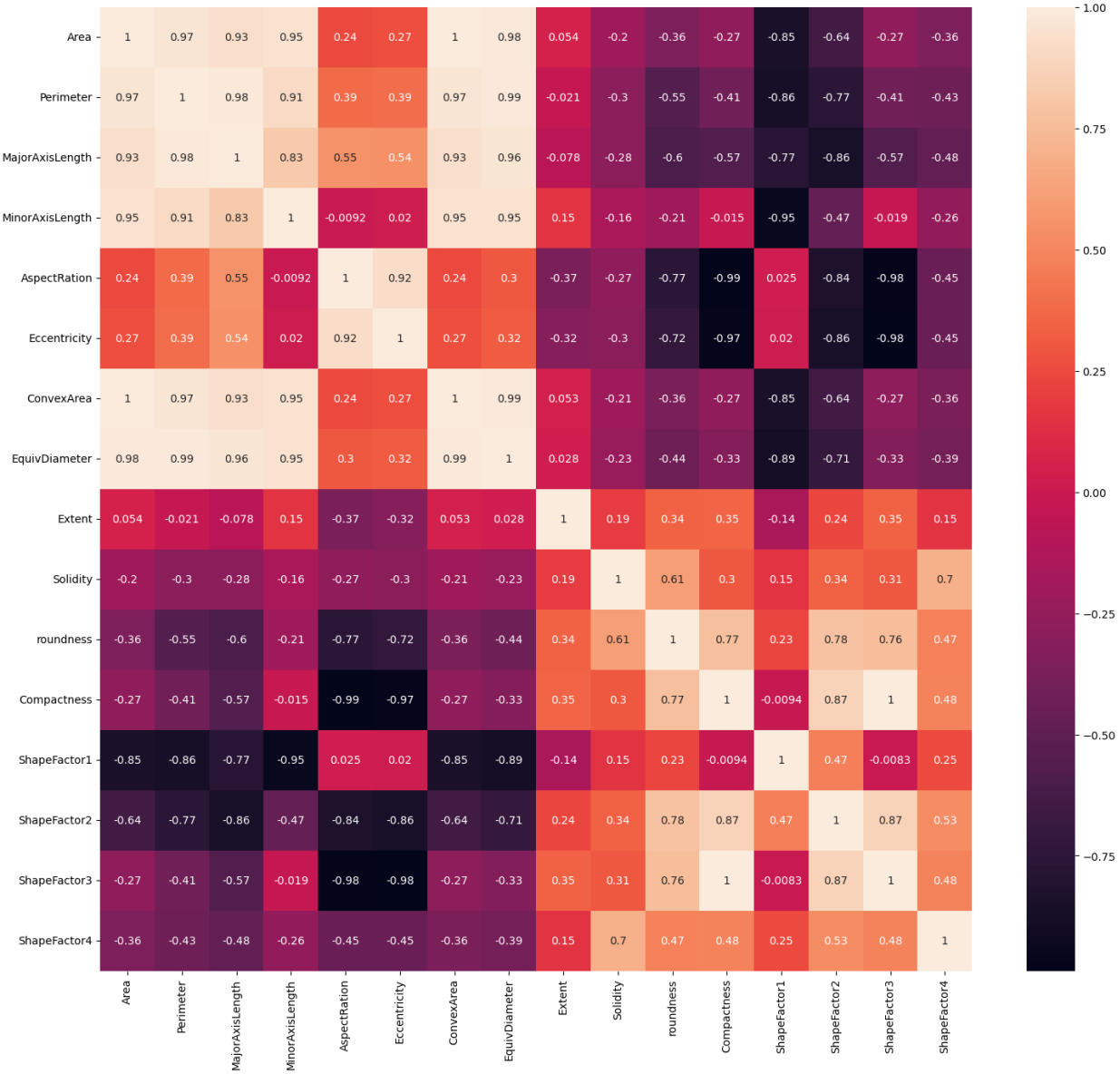
In [22]:
```
plt.figure(figsize= (10,8))
df.plot.scatter(x='roundness', y='Compactness')
plt.show()
```

```
<Figure size 1000x800 with 0 Axes>
```



Find high correlated columns

In [12]:
```
plt.figure(figsize=(18,16))
sns.heatmap(data= df.corr(), annot= True)
plt.show()
```



Use Label Encoder for convert Categorical into Numerical data

In [27]:
```
le= LabelEncoder()
df['Class'] =le.fit_transform(df['Class'])
```

In [28]:
```
df.head()
```

Out[28]:

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	ConvexArea	EquivDiame
0	28395	610.291	208.178117	173.888747	1.197191	0.549812	28715	190.141
1	28734	638.018	200.524796	182.734419	1.097356	0.411785	29172	191.272
2	29380	624.110	212.826130	175.931143	1.209713	0.562727	29690	193.410
3	30008	645.884	210.557999	182.516516	1.153638	0.498616	30724	195.467
4	30140	620.134	201.847882	190.279279	1.060798	0.333680	30417	195.896

some of features are highly correlated that reason we drop those columns

1- Area, Perimeter, MajorAxislength, MinorAxisLength, AsceptionRatio, ConvexArea, ShapFactor2, ShapeFactor3,compactness

In [29]:
```
df.drop(['Area','Perimeter','MajorAxisLength','MinorAxisLength','AspectRation','ConvexAr
```

In [30]:
```
# After drop the columns
df.head()
```

Out[30]:

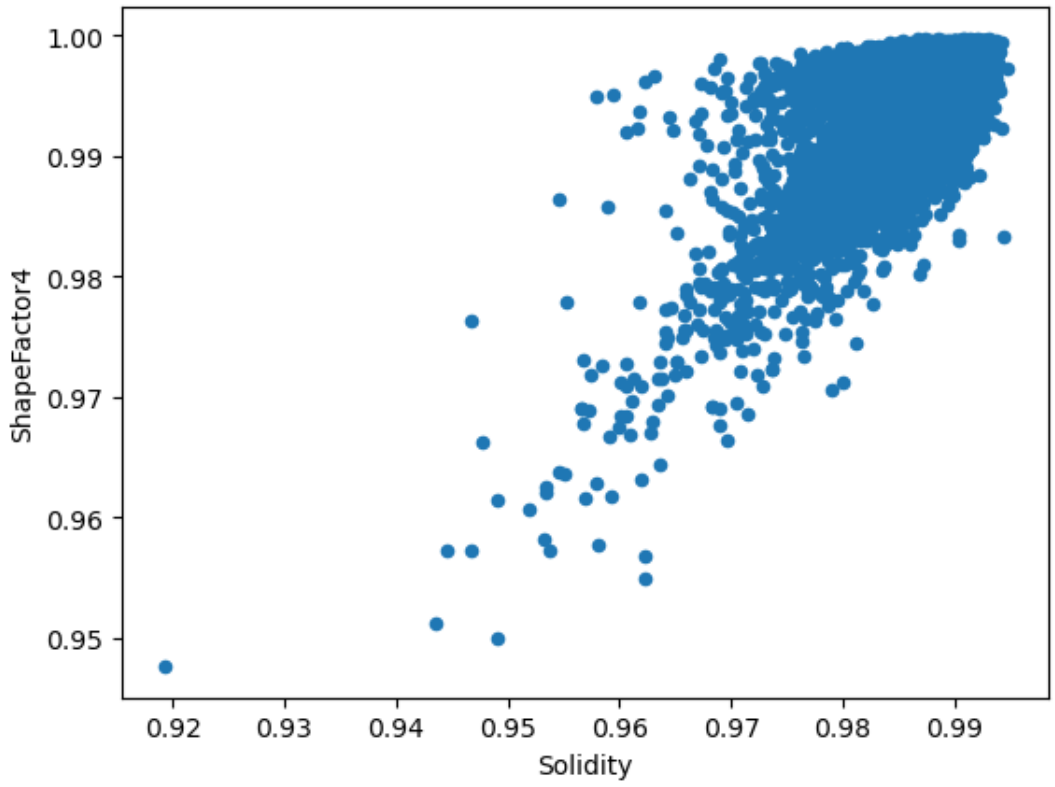
	Eccentricity	EquivDiameter	Extent	Solidity	roundness	ShapeFactor1	ShapeFactor4	Class
0	0.549812	190.141097	0.763923	0.988856	0.958027	0.007332	0.998724	5
1	0.411785	191.272751	0.783968	0.984986	0.887034	0.006979	0.998430	5
2	0.562727	193.410904	0.778113	0.989559	0.947849	0.007244	0.999066	5
3	0.498616	195.467062	0.782681	0.976696	0.903936	0.007017	0.994199	5
4	0.333680	195.896503	0.773098	0.990893	0.984877	0.006697	0.999166	5

In [58]:
```
# After Drop the columns
df.describe()
```

Out[58]:

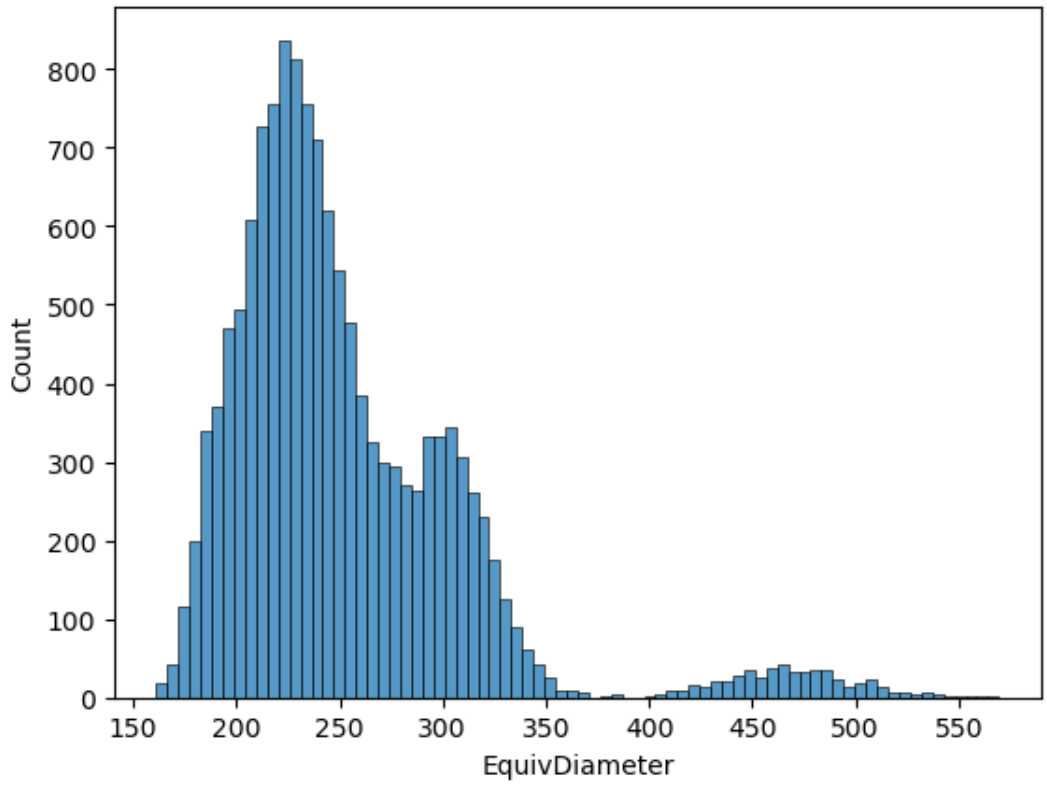
	Eccentricity	EquivDiameter	Extent	Solidity	roundness	ShapeFactor1	ShapeFactor4
count	13611.000000	13611.000000	13611.000000	13611.000000	13611.000000	13611.000000	13611.000000
mean	0.750895	253.064220	0.749733	0.987143	0.873282	0.006564	0.995063
std	0.092002	59.177120	0.049086	0.004660	0.059520	0.001128	0.004366
min	0.218951	161.243764	0.555315	0.919246	0.489618	0.002778	0.947687
25%	0.715928	215.068003	0.718634	0.985670	0.832096	0.005900	0.993703
50%	0.764441	238.438026	0.759859	0.988283	0.883157	0.006645	0.996386
75%	0.810466	279.446467	0.786851	0.990013	0.916869	0.007271	0.997883
max	0.911423	569.374358	0.866195	0.994677	0.990685	0.010451	0.999733

In [60]:
```
# plot the scatter plot between Solidity and ShapeFactor4
df.plot.scatter(x= 'Solidity' ,y= 'ShapeFactor4')
plt.show()
```

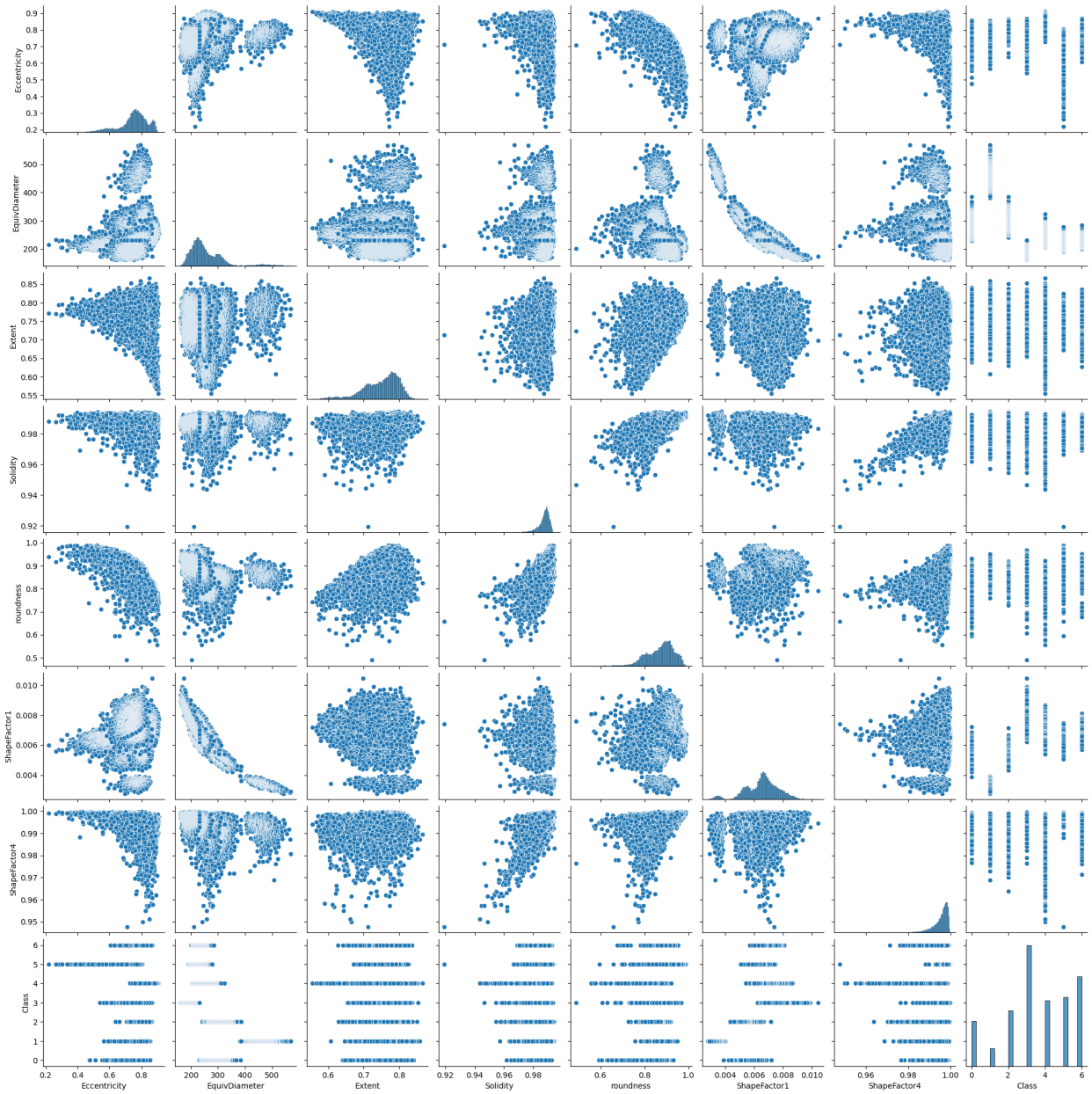


In [31]:
```
# create a histplot for EquivDiameter
sns.histplot(df['EquivDiameter'])
```

Out[31]: <AxesSubplot:xlabel='EquivDiameter', ylabel='Count'>



In [32]:
```
# Create a pair plot for Data
sns.pairplot(data= df)
plt.show()
```



Split the data into Training and testing Data

In [33]:
```
X= df.iloc[:,:-1]
y= df.iloc[:,-1]
```

In [36]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=42)
```

In [37]:
```
print('X_train: ',X_train.shape)
print('X_test: ',X_test.shape)
print('y_train: ',y_train.shape)
print('y_test: ',y_test.shape)
```

```
X_train:  (9527, 7)
X_test:  (4084, 7)
y_train:  (9527,)
y_test:  (4084,)
```

In [38]:
```
y_train= y_train.values.reshape(y_train.shape[0],1)
y_test= y_test.values.reshape(y_test.shape[0],1)
```

In [41]:
```
sc= StandardScaler()
X_train_scaled= sc.fit_transform(X_train)
X_test_scaled= sc.fit_transform(X_test)
```

In [45]:
```
# Use SVC for predication
from sklearn.svm import SVC
clf= SVC(kernel= 'rbf', random_state=0)
clf.fit(X_train_scaled ,y_train)
y_pred= clf.predict(X_test_scaled)
```

```
C:\Users\hp\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversio
nWarning: A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [47]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
Accuracy= accuracy_score(y_test, y_pred)
Accuracy
```

Out[47]:
```
0.9324191968658179
```

In [49]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [50]:
```
DTC= DecisionTreeClassifier()
DTC.fit(X_train_scaled ,y_train)
```

Out[50]:
```
DecisionTreeClassifier()
```

In [51]:
```
y_pred2 =DTC.predict(X_test_scaled)
y_pred2
```

Out[51]:
```
array([5, 0, 5, ..., 4, 5, 0])
```

In [53]:
```
from sklearn import metrics
rmse= np.sqrt(metrics.mean_squared_error(y_test,y_pred2))
rmse
```

Out[53]:
```
0.8893206862857894
```

In [54]:
```
y_pred_df= pd.DataFrame(y_pred2)
y_pred_df
```

Out[54]:

	0
0	5
1	0
2	5
3	5
4	3
...	...
4079	5
4080	3
4081	4
4082	5
4083	0

4084 rows × 1 columns

In [55]:
```
y_pred_df['actual'] = y_test
```

In [57]:
```
y_pred_df.head()
```

Out[57]:

	0	actual
0	5	5
1	0	0
2	5	5
3	5	5
4	3	3

Insights

1- Maximum ShapeFactor4 is 0.999 or approx 1.

2- Most of the Data are Correlated.

3- We use the SVC for Classification that gave accuracy 93.24%.

4- We use Decision Tree Classifier that gave rmse value .89 is good sign to data is predicted value.

In [ ]: