Importing Libraries

```
import pandas as pd
from numpy import asarray
from sklearn.datasets import make_classification
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from pandas import read_csv
import matplotlib.pyplot as plt
# df = pd.read_csv(r"C:\Users\hp\Downloads\DryBeanDataset\DryBeanDataset\Dry_Bean_Data

```

```

```

Loading the dataset

```
# If you've got the file in local drive ignore this step
! gdown --id 1JxNUhdP4fby1QlTdMnUbSz9iZjX4bPTs
```

```
/usr/local/lib/python3.7/dist-packages/gdown/cli.py:131: FutureWarning: Option `--id`
  category=FutureWarning,
Downloading...
From: https://drive.google.com/uc?id=1JxNUhdP4fby1QlTdMnUbSz9iZjX4bPTs
To: /content/Dry_Bean.csv
100% 2.48M/2.48M [00:00<00:00, 181MB/s]
```

```
# Reading the data in csv
df = pd.read_csv('Dry_Bean.csv')
df.head()
```

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity
0	28395	610.291	208.178117	173.888747	1.197191	0.549812
1	28734	638.018	200.524796	182.734419	1.097356	0.411785
2	29380	624.110	212.826130	175.931143	1.209713	0.562727
3	30008	645.884	210.557999	182.516516	1.153638	0.498616
4	30140	620.134	201.847882	190.279279	1.060798	0.333680

```
# Data information about columns and rows
# Displaying infromation for clear underatsanding
df.info()
print(df.columns)
print(df.index)
print(df.values)
print(df.values)
```

```
<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
Index(['Area', 'Perimeter', 'MajorAxisLength', 'MinorAxisLength',
       'AspectRation', 'Eccentricity', 'ConvexArea', 'EquivDiameter', 'Extent',
       'Solidity', 'roundness', 'Compactness', 'ShapeFactor1', 'ShapeFactor2',
       'ShapeFactor3', 'ShapeFactor4', 'Class'],
      dtype='object')
RangeIndex(start=0, stop=13611, step=1)
[[28395 610.291 208.1781167 ... 0.834222388 0.998723889 'SEKER']
 [28734 638.018 200.5247957 ... 0.909850506 0.998430331 'SEKER']
 [29380 624.11 212.8261299 ... 0.825870617 0.999066137 'SEKER']
 ...
 [42139 759.321 281.5399279 ... 0.676884164 0.996767264 'DERMASON']
 [42147 763.779 283.3826364 ... 0.668236684 0.99522242 'DERMASON']
 [42159 772.237 295.142741 ... 0.616220592 0.998179623 'DERMASON']]
```

```
# Data cleaning
print(df.isnull().sum())
df = df.dropna(axis=0)
```

```
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
```

```
df.head()
```

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity
0	28395	610.291	208.178117	173.888747	1.197191	0.549812
1	28734	638.018	200.524796	182.734419	1.097356	0.411785
2	29380	624.110	212.826130	175.931143	1.209713	0.562727
3	30008	645.884	210.557999	182.516516	1.153638	0.498616
4	30140	620.134	201.847882	190.279279	1.060798	0.333680

```
# tell about the class
type(df['Class'])

```

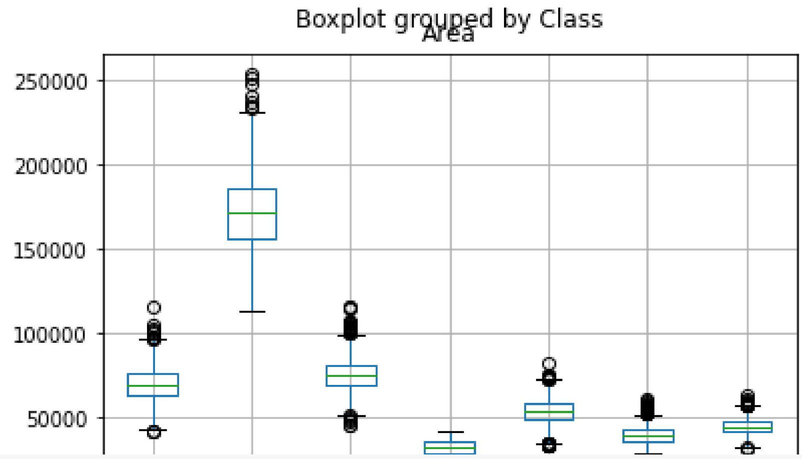
```
pandas.core.series.Series
```

```
# Tell the information about dataset classes
df.describe()
```

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	E
count	13611.000000	13611.000000	13611.000000	13611.000000	13611.000000	
mean	53048.284549	855.283459	320.141867	202.270714	1.583242	
std	29324.095717	214.289696	85.694186	44.970091	0.246678	
min	20420.000000	524.736000	183.601165	122.512653	1.024868	
25%	36328.000000	703.523500	253.303633	175.848170	1.432307	
50%	44652.000000	794.941000	296.883367	192.431733	1.551124	
75%	61332.000000	977.213000	376.495012	217.031741	1.707109	
max	254616.000000	1985.370000	738.860154	460.198497	2.430306	

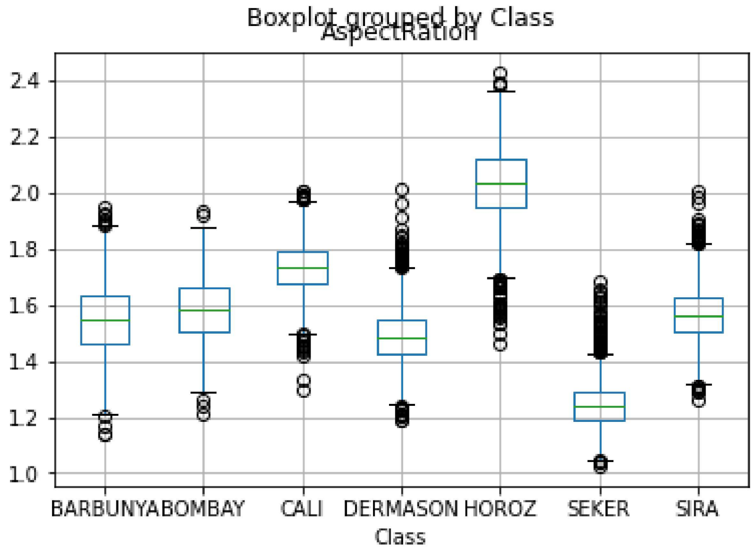
```
boxplot = df.boxplot(column=['Area'], by='Class')
# Tried the box plot. learned from tableau
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/cbook/__init__.py:1376: VisibleDepr
  X = np.atleast_1d(X.T if isinstance(X, np.ndarray) else np.asarray(X))
```



```
boxplot = df.boxplot(column=['AspectRation'], by='Class')
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/cbook/__init__.py:1376: VisibleDepr
  X = np.atleast_1d(X.T if isinstance(X, np.ndarray) else np.asarray(X))
```

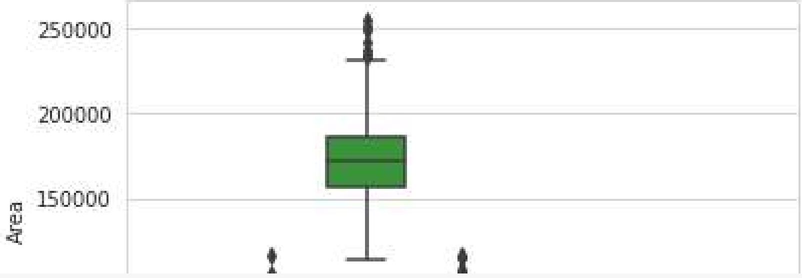


Data Exploration

```
# importing seaborn library
import seaborn as sns
```

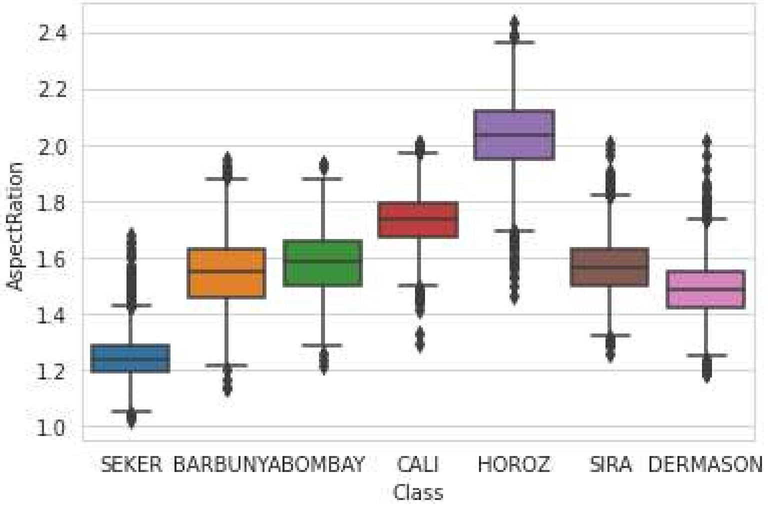
```
sns.set_style("whitegrid")
sns.boxplot(x = 'Class', y = 'Area', data = df)
# Adding features to it as well understanding it
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f54b3f60b50>
```

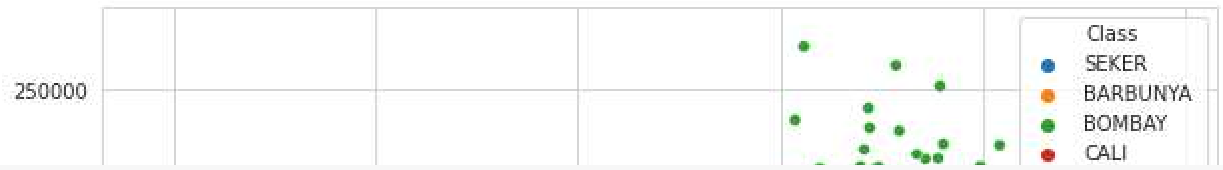


```
sns.boxplot(x = 'Class', y = 'AspectRation', data = df)
```

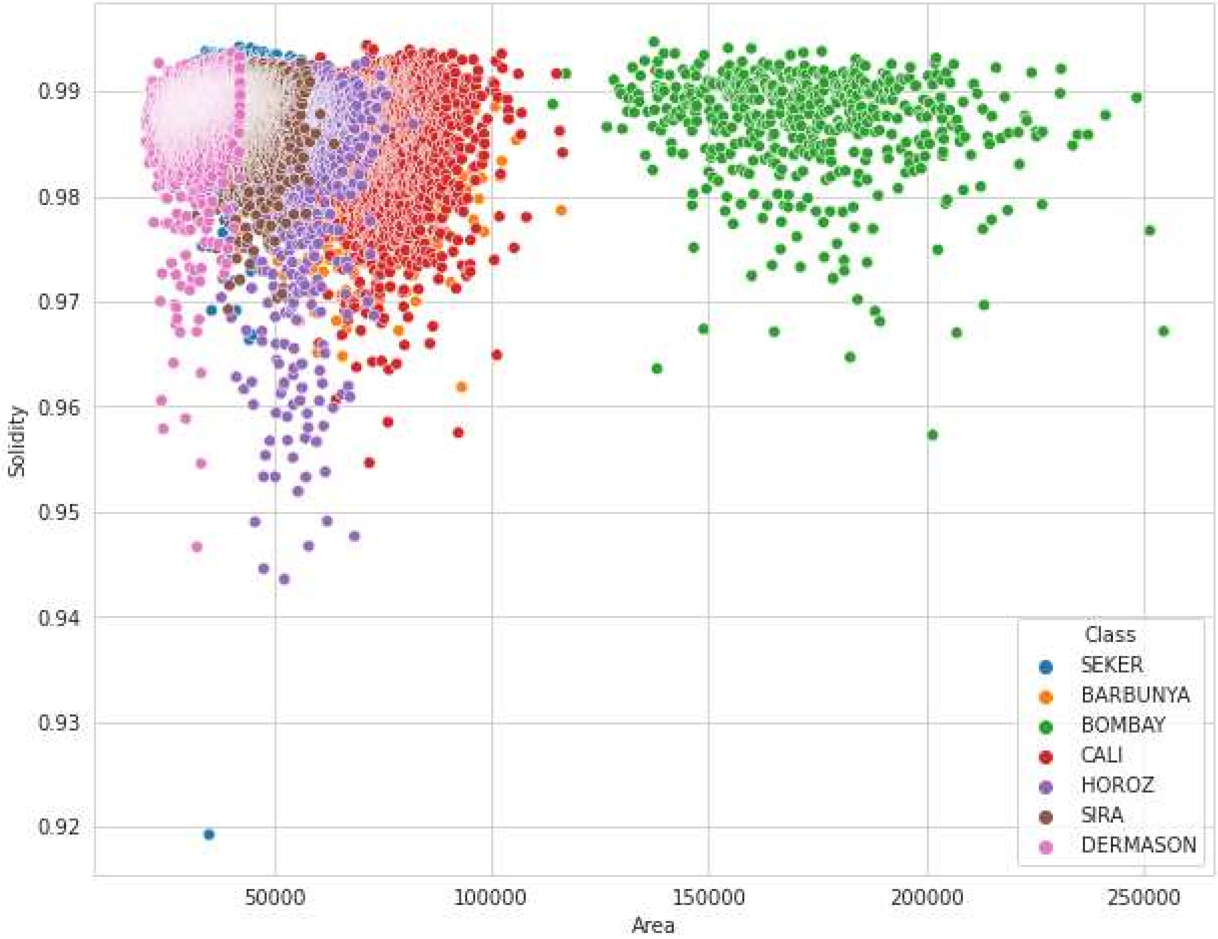
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f54b3667790>
```



```
# Trying various methods to identify the correctness
plt.figure(figsize=(10,8))
sns.scatterplot(x=df.roundness , y=df['ConvexArea'], hue=df.Class )
plt.show()
# tried scatter plot representation to check the code
```



```
plt.figure(figsize=(10,8))
sns.scatterplot(x=df.Area , y=df['Solidity'], hue=df.Class )
plt.show()
```



```
```

Altair Library

Altair is a declarative statistical visualization library for Python, based on Vega and Vega-Lite.

Altair offers a powerful and concise visualization grammar that enables you to build a wide range of statistical visualizations quickly. Here is an example of using the Altair API to quickly visualize a dataset with an interactive scatter plot:

Source: https://altair-viz.github.io/getting_started/overview.html
https://altair-viz.github.io/gallery/boxplot.html

```
# importing altair library
```

```
import altair as alt
```

```
## dropping an unused column
df2 = df.drop(['Perimeter','MajorAxisLength','MinorAxisLength','Eccentricity','EquivDi
```

Count Unique Values

```
# By using drop_duplicates()
count = df2.Class.drop_duplicates().size
print("Unique values count : "+ str(count))
```

```
Unique values count : 7
```

```
print(df2.Class.value_counts())
```

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

Looking ar the data we need to get equal number of rows for all categories, therefore chossing the respective categories 522 rows.

```
df3 = df2.groupby("Class").head(522)
```

```
print(df3[["Class"]].value_counts())
```

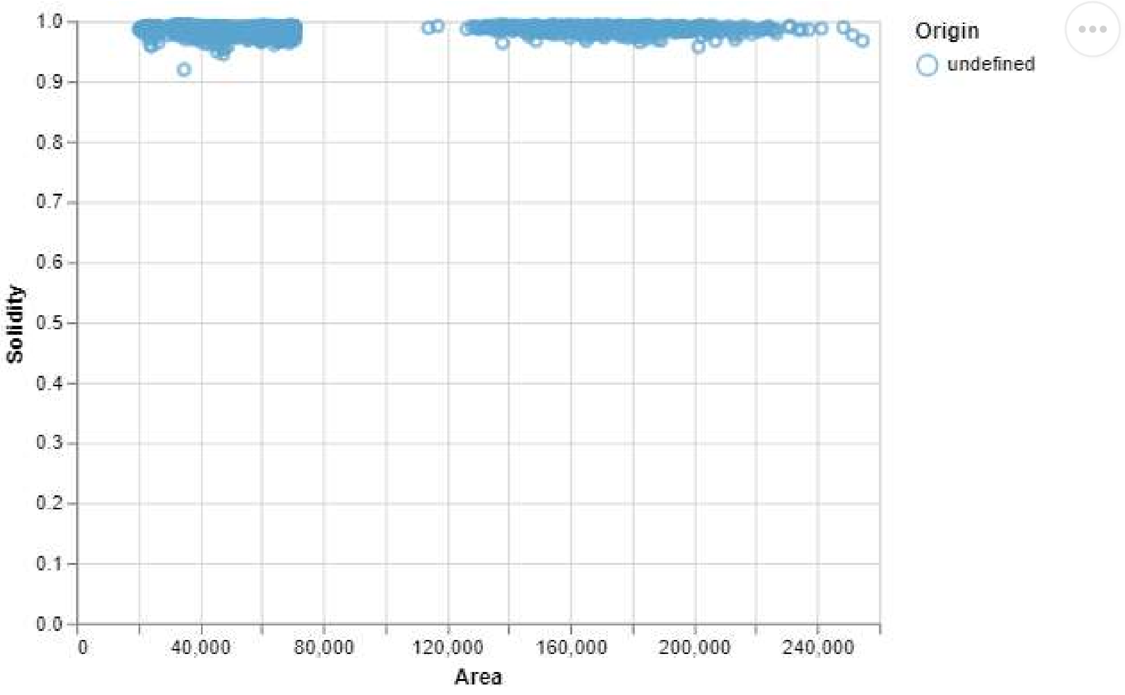
```
Class
BARBUNYA    522
BOMBAY      522
CALI        522
DERMASON    522
HOROZ       522
SEKER       522
SIRA        522
dtype: int64
```

```
df3.columns
```

```
Index(['Area', 'AspectRation', 'ConvexArea', 'Solidity', 'roundness', 'Class'],
dtype='object')
```

https://altair-viz.github.io/user_guide/encoding.html

```
alt.Chart(df3).mark_point().encode(
    x='Area:Q',
    y='Solidity:Q',
    color='Origin:O',
    shape='Origin:N'
)
```

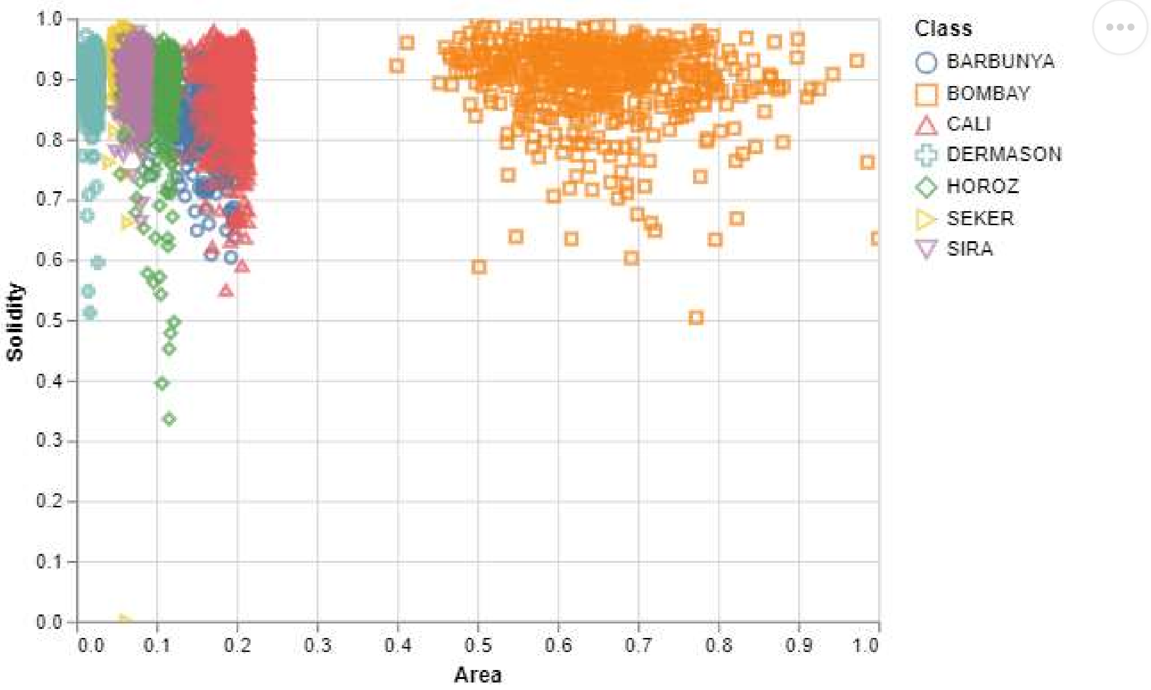


😞 The data representation is not good ! The reason being data is not normalized as we limited data to certain number of rows only ! Therefore we need to use the technique of ***NORMALIZATION***.

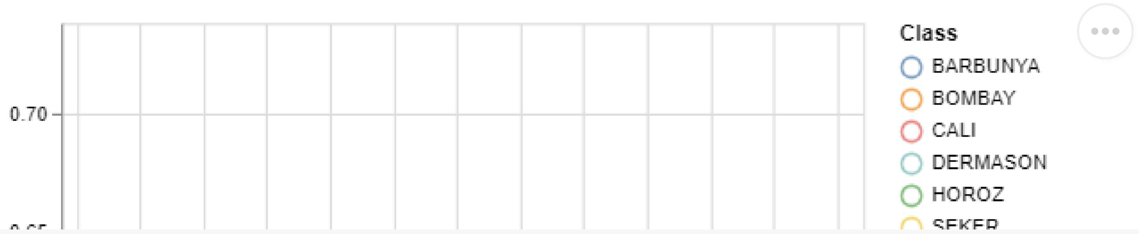
https://www.geeksforgeeks.org/data-normalization-with-pandas/
https://www.datacamp.com/tutorial/altair-in-python
https://towardsdatascience.com/data-normalization-with-pandas-and-scikit-learn-7c1cc6ed6475

```
# Normalizing data
# copy the data
df_min_max_scaled = df3.copy()

# apply normalization techniques
for column in df_min_max_scaled.columns:
    if column == 'Class':
        continue
    df_min_max_scaled[column] = (df_min_max_scaled[column] - df_min_max_scaled[column]

# view normalized data
# print(df_min_max_scaled)
```

```
alt.Chart(df_min_max_scaled).mark_point().encode(
    x='Area:Q',
    y='Solidity:Q',
    color='Class:N',
    shape='Class:N',
    tooltip = [alt.Tooltip('Class'),
               alt.Tooltip('Solidity'),
               alt.Tooltip('Area')
              ]
).interactive()
# Represented in a format as that of our prototype.
# As stated in class perform one of the prototype
```

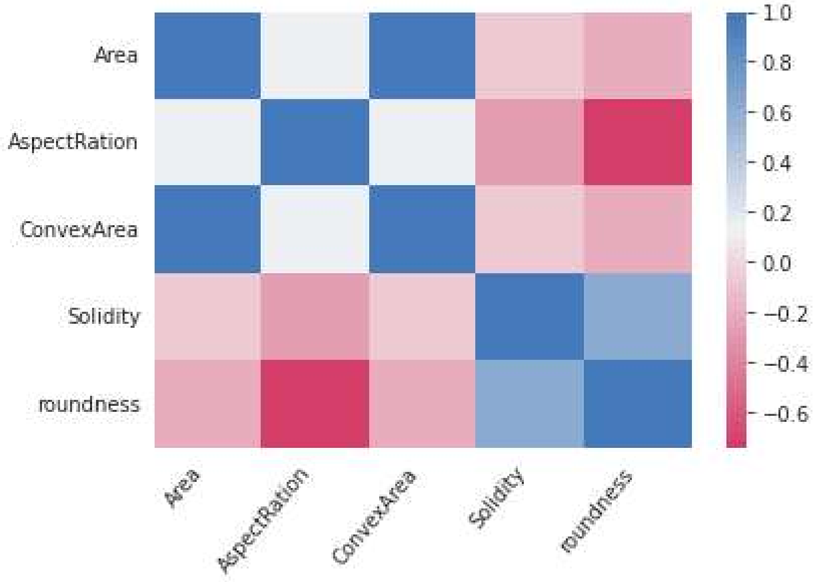


```
alt.Chart(df_min_max_scaled).mark_point().encode(
    x='Area:Q',
    y='Solidity:Q',
    color='Class:N',
    shape='Origin:N',
    tooltip = [alt.Tooltip('Class'),
               alt.Tooltip('Solidity'),
               alt.Tooltip('Area')
              ]
).interactive()
# Perfomred the
```



```
#  we are finding a corelation between two attributes
# through a co-relation fxn
print(df_min_max_scaled.corr())
correlationMatrix=df_min_max_scaled.corr(method='pearson')
ax = sns.heatmap(
  correlationMatrix
#    ,  min=-1, vmax=1, center=0,
  ,cmap=sns.diverging_palette(0, 250, n=50)
#    ,  square=True
)

ax.set_xticklabels (
    ax.get_xticklabels (),
    rotation=50,
    horizontalalignment='right'
);
# Through the matrix we can find thew inter-relation of the data set but as per our pr
# attributes we require are represented in a better way through graph.
```

```
                 Area  AspectRation  ConvexArea  Solidity  roundness
Area          1.000000      0.136519    0.999959 -0.050597  -0.189505
AspectRation  0.136519      1.000000    0.137894 -0.257164  -0.745264
ConvexArea    0.999959      0.137894    1.000000 -0.058001  -0.193057
Solidity     -0.050597     -0.257164   -0.058001  1.000000   0.617543
roundness    -0.189505     -0.745264   -0.193057  0.617543   1.000000
```


Area
AspectRation
ConvexArea
Solidity
roundness
Area
AspectRation
ConvexArea
Solidity
roundness