Importing the Dependencies

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import ExtraTreesClassifier
from sklearn import preprocessing
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report,accuracy_score
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.model_selection import RandomizedSearchCV
```

In [2]:
```
# Data collection & Analysis
```

In [3]:
```
data = pd.read_csv('DATA.csv')
data.head()
```

Out[3]:

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	ConvexArea	Equivd
0	28395	610.291	208.178117	173.888747	1.197191	0.549812	28715	19
1	28734	638.018	200.524796	182.734419	1.097356	0.411785	29172	19
2	29380	624.110	212.826130	175.931143	1.209713	0.562727	29690	19
3	30008	645.884	210.557999	182.516516	1.153638	0.498616	30724	19
4	30140	620.134	201.847882	190.279279	1.060798	0.333680	30417	19

In [4]:
```
data.columns
```

Out[4]:
```
Index(['Area', 'Perimeter', 'MajorAxisLength', 'MinorAxisLength',
       'AspectRation', 'Eccentricity', 'ConvexArea', 'EquivDiameter', 'Extent',
       'Solidity', 'roundness', 'Compactness', 'ShapeFactor1', 'ShapeFactor2',
       'ShapeFactor3', 'ShapeFactor4', 'Class'],
      dtype='object')
```

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

Out[5]:
```
(13611, 17)
```

In [6]:
```
data.size
```

Out[6]:
```
231387
```

In [7]:
```
data.describe()
```

Out[7]:

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity
count	13611.000000	13611.000000	13611.000000	13611.000000	13611.000000	13611.000000
mean	53048.284549	855.283459	320.141867	202.270714	1.583242	0.750895
std	29324.095717	214.289696	85.694186	44.970091	0.246578	0.092002
min	20420.000000	524.736000	183.601165	122.512653	1.024868	0.218951
25%	36328.000000	703.523500	253.303633	175.848170	1.432307	0.715928
50%	44652.000000	794.941000	296.883367	192.431733	1.551124	0.764441
75%	61332.000000	977.213000	376.495012	217.031741	1.707109	0.810466
max	254616.000000	1985.370000	738.860154	460.198497	2.430306	0.911423

From this data overview we can see that:

The dataset has 13611 rows and 17 columns All columns are numerical except "Class" column which will be our target There are no NULL values in the data thanks to which we have less work to do By looking at the description of the data we can come to the conclusion that our data will need standardization

In [8]:
```
data.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
```

In [9]:
```
data.isnull().sum()
```

Out[9]:
```
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
```

Target Info

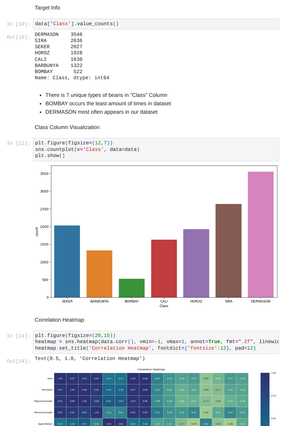
In [10]:
```
data['Class'].value_counts()
```

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

- There is 7 unique types of beans in "Class" Column
- BOMBAY occurs the least amount of times in dataset
- DERMASON most often appears in our dataset

Class Column Visualization

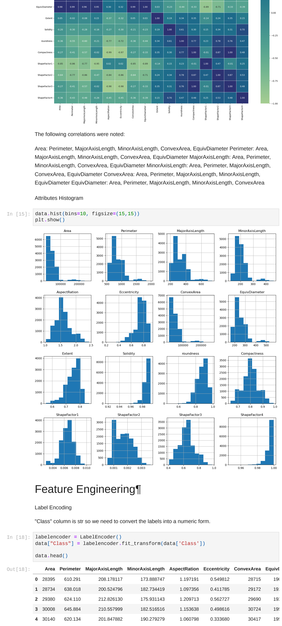
In [11]:
```
plt.figure(figsize=(12,7))
sns.countplot(x='Class', data=data)
plt.show()
```



Correlation Heatmap

In [12]:
```
plt.figure(figsize=(20,15))
heatmap = sns.heatmap(data.corr(), vmin=-1, vmax=1, annot=True, fmt='.2f', linewidths=2)
heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':12}, pad=12)
```

Out[12]:
```
Text(0.5, 1.0, 'Correlation Heatmap')
```

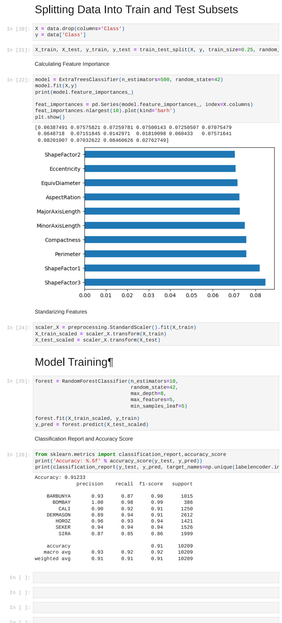


The following correlations were noted:

Area: Perimeter, MajorAxisLength, MinorAxisLength, ConvexArea, EquivDiameter Perimeter: Area, MajorAxisLength, MinorAxisLength, ConvexArea, EquivDiameter MajorAxisLength: Area, Perimeter, MinorAxisLength, ConvexArea, EquivDiameter MinorAxisLength: Area, Perimeter, MajorAxisLength, ConvexArea, EquivDiameter ConvexArea: Area, Perimeter, MajorAxisLength, MinorAxisLength, EquivDiameter EquivDiameter: Area, Perimeter, MajorAxisLength, MinorAxisLength, ConvexArea

Attributes Histogram

In [13]:
```
data.hist(bins=10, figsize=(15,15))
plt.show()
```



Feature Engineering¶

Label Encoding

"Class" column is str so we need to convert the labels into a numeric form.

In [14]:
```
labelencoder = LabelEncoder()
data['Class'] = labelencoder.fit_transform(data['Class'])

data.head()
```

Out[14]:

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	ConvexArea	Equivd
0	28395	610.291	208.178117	173.888747	1.197191	0.549812	28715	19
1	28734	638.018	200.524796	182.734419	1.097356	0.411785	29172	19
2	29380	624.110	212.826130	175.931143	1.209713	0.562727	29690	19
3	30008	645.884	210.557999	182.516516	1.153638	0.498616	30724	19
4	30140	620.134	201.847882	190.279279	1.060798	0.333680	30417	19

Splitting Data Into Train and Test Subsets

In [15]:
```
X = data.drop(columns='Class')
y = data['Class']
```

In [16]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.25, random_state=42)
```

Calculating Feature Importance

In [17]:
```
model = ExtraTreesClassifier(n_estimators=500, random_state=42)
model.fit(X,y)
print(model.feature_importances_)

feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.show()
```

```
[0.06387493 0.07575821 0.07259781 0.07560141 0.07259507 0.07075479
 0.06458718 0.07151845 0.0143071  0.01810998 0.068433   0.07571841
 0.08201007 0.07832622 0.08460626 0.02762749]
```


Standarizing Features

In [18]:
```
scaler_X = preprocessing.StandardScaler().fit(X_train)
X_train_scaled = scaler_X.transform(X_train)
X_test_scaled = scaler_X.transform(X_test)
```

Model Training¶

In [19]:
```
forest = RandomForestClassifier(n_estimators=10,
                                random_state=42,
                                max_depth=8,
                                max_features=5,
                                min_samples_leaf=5)

forest.fit(X_train_scaled, y_train)
y_pred = forest.predict(X_test_scaled)
```

Classification Report and Accuracy Score

In [20]:
```
from sklearn.metrics import classification_report,accuracy_score
print('Accuracy: %.5f' % accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=np.unique(labelencoder.inverse_transform(y))))
```

```
Accuracy: 0.91233
              precision    recall  f1-score   support

    BARBUNYA       0.93      0.87      0.90      1015
      BOMBAY       1.00      0.98      0.99       386
        CALI       0.90      0.92      0.91      1250
    DERMASON       0.89      0.94      0.91      2612
       HOROZ       0.96      0.93      0.94      1421
       SEKER       0.94      0.94      0.94      1525
        SIRA       0.87      0.85      0.86      1999

    accuracy                           0.91     10209
   macro avg       0.93      0.92      0.92     10209
weighted avg       0.91      0.91      0.91     10209
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```