# Apply statistical modelling techniques to classify dry beans (part-2)

In [251…
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [253…
```
df=pd.read_csv(r"C:\Users\jayant soni\Downloads\Dry_Bean.csv.zip")
df
```

Out[253]:

| | Area | Perimeter | MajorAxisLength | MinorAxisLength | AspectRation | Eccentricity | ConvexArea | EquivDiameter | Extent | Solidity | roundness | Compactness | ShapeFactor1 | ShapeFactor2 | ShapeFactor3 | ShapeFactor4 | Class |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 28395 | 610.291 | 208.178117 | 173.888747 | 1.197191 | 0.549812 | 28715 | 190.141097 | 0.763923 | 0.988856 | 0.958027 | 0.913358 | 0.007332 | 0.003147 | 0.834222 | 0.998724 | SEKER |
| **1** | 28734 | 638.018 | 200.524796 | 182.734419 | 1.097356 | 0.411785 | 29172 | 191.272751 | 0.783968 | 0.984986 | 0.887034 | 0.953861 | 0.006979 | 0.003564 | 0.909851 | 0.998430 | SEKER |
| **2** | 29380 | 624.110 | 212.826130 | 175.931143 | 1.209713 | 0.562727 | 29690 | 193.410904 | 0.778113 | 0.989559 | 0.947849 | 0.908774 | 0.007244 | 0.003048 | 0.825871 | 0.999066 | SEKER |
| **3** | 30008 | 645.884 | 210.557999 | 182.516516 | 1.153638 | 0.498616 | 30724 | 195.467062 | 0.782681 | 0.976696 | 0.903936 | 0.928329 | 0.007017 | 0.003215 | 0.861794 | 0.994199 | SEKER |
| **4** | 30140 | 620.134 | 201.847882 | 190.279279 | 1.060798 | 0.333680 | 30417 | 195.896503 | 0.773098 | 0.990893 | 0.984877 | 0.970516 | 0.006697 | 0.003665 | 0.941900 | 0.999166 | SEKER |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **13606** | 42097 | 759.696 | 288.721612 | 185.944705 | 1.552728 | 0.765002 | 42508 | 231.515799 | 0.714574 | 0.990331 | 0.916603 | 0.801865 | 0.006858 | 0.001749 | 0.642988 | 0.998385 | DERMASON |
| **13607** | 42101 | 757.499 | 281.576392 | 190.713136 | 1.476439 | 0.735702 | 42494 | 231.526798 | 0.799943 | 0.990752 | 0.922015 | 0.822252 | 0.006688 | 0.001886 | 0.676099 | 0.998219 | DERMASON |
| **13608** | 42139 | 759.321 | 281.539928 | 191.187979 | 1.472582 | 0.734065 | 42569 | 231.631261 | 0.729932 | 0.989899 | 0.918424 | 0.822730 | 0.006681 | 0.001888 | 0.676884 | 0.996767 | DERMASON |
| **13609** | 42147 | 763.779 | 283.382636 | 190.275731 | 1.489326 | 0.741055 | 42667 | 231.653247 | 0.705389 | 0.987813 | 0.907906 | 0.817457 | 0.006724 | 0.001852 | 0.668237 | 0.995222 | DERMASON |
| **13610** | 42159 | 772.237 | 295.142741 | 182.204716 | 1.619841 | 0.786693 | 42600 | 231.686223 | 0.788962 | 0.989648 | 0.888380 | 0.784997 | 0.007001 | 0.001640 | 0.616221 | 0.998180 | DERMASON |

13611 rows × 17 columns

In [254…
```
df.describe()
```

Out[254]:

| | Area | Perimeter | MajorAxisLength | MinorAxisLength | AspectRation | Eccentricity | ConvexArea | EquivDiameter | Extent | Solidity | roundness | Compactness | ShapeFactor1 | ShapeFactor2 | ShapeFactor3 | ShapeFactor4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 13611.000000 | 13611.000000 | 13611.000000 | 13611.000000 | 13611.000000 | 13611.000000 | 13611.000000 | 13611.000000 | 13611.000000 | 13611.000000 | 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 | 53768.200206 | 253.064220 | 0.749733 | 0.987143 | 0.873282 | 0.799864 | 0.006564 | 0.001716 | 0.643590 | 0.995063 |
| **std** | 29324.095717 | 214.289696 | 85.694186 | 44.970091 | 0.246678 | 0.092002 | 29774.915817 | 59.177120 | 0.049086 | 0.004660 | 0.059520 | 0.061713 | 0.001128 | 0.000596 | 0.098996 | 0.004366 |
| **min** | 20420.000000 | 524.736000 | 183.601165 | 122.512653 | 1.024868 | 0.218951 | 20684.000000 | 161.243764 | 0.555315 | 0.919246 | 0.489618 | 0.640577 | 0.002778 | 0.000564 | 0.410339 | 0.947687 |
| **25%** | 36328.000000 | 703.523500 | 253.303633 | 175.848170 | 1.432307 | 0.715928 | 36714.500000 | 215.068003 | 0.718634 | 0.985670 | 0.832096 | 0.762469 | 0.005900 | 0.001154 | 0.581359 | 0.993703 |
| **50%** | 44652.000000 | 794.941000 | 296.883367 | 192.431733 | 1.551124 | 0.764441 | 45178.000000 | 238.438026 | 0.759859 | 0.988283 | 0.883157 | 0.801277 | 0.006645 | 0.001694 | 0.642044 | 0.996386 |
| **75%** | 61332.000000 | 977.213000 | 376.495012 | 217.031741 | 1.707109 | 0.810466 | 62294.000000 | 279.446467 | 0.786851 | 0.990013 | 0.916869 | 0.834270 | 0.007271 | 0.002170 | 0.696006 | 0.997883 |
| **max** | 254616.000000 | 1985.370000 | 738.860154 | 460.198497 | 2.430306 | 0.911423 | 263261.000000 | 569.374358 | 0.866195 | 0.994677 | 0.990685 | 0.987303 | 0.010451 | 0.003665 | 0.974767 | 0.999733 |

In [255…
```
df.isnull().sum()
```

Out[255]:
```
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 [256…
```
df.shape
```

Out[256]:
```
(13611, 17)
```

In [257…
```
df['Class'].unique()
```

Out[257]:
```
array(['SEKER', 'BARBUNYA', 'BOMBAY', 'CALI', 'HOROZ', 'SIRA', 'DERMASON'],
      dtype=object)
```

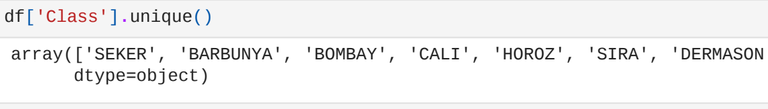
In [258…
```
corr=df.corr()
corr
```

Out[258]:

| | Area | Perimeter | MajorAxisLength | MinorAxisLength | AspectRation | Eccentricity | ConvexArea | EquivDiameter | Extent | Solidity | roundness | Compactness | ShapeFactor1 | ShapeFactor2 | ShapeFactor3 | ShapeFactor4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Area** | 1.000000 | 0.966722 | 0.931834 | 0.951602 | 0.241735 | 0.267481 | 0.999939 | 0.984968 | 0.054345 | -0.196585 | -0.357530 | -0.268067 | -0.847958 | -0.639291 | -0.272145 | -0.355721 |
| **Perimeter** | 0.966722 | 1.000000 | 0.977338 | 0.913179 | 0.385276 | 0.391066 | 0.967689 | 0.991380 | -0.021160 | -0.303970 | -0.547647 | -0.406857 | -0.864623 | -0.767592 | -0.408435 | -0.429310 |
| **MajorAxisLength** | 0.931834 | 0.977338 | 1.000000 | 0.826052 | 0.550335 | 0.541972 | 0.932607 | 0.961733 | -0.078062 | -0.284302 | -0.596358 | -0.568377 | -0.773609 | -0.859238 | -0.568185 | -0.482527 |
| **MinorAxisLength** | 0.951602 | 0.913179 | 0.826052 | 1.000000 | -0.009161 | 0.019574 | 0.951339 | 0.948539 | 0.145957 | -0.155831 | -0.210344 | -0.015066 | -0.947204 | -0.471347 | -0.019326 | -0.263749 |
| **AspectRation** | 0.241735 | 0.385276 | 0.550335 | -0.009161 | 1.000000 | 0.924293 | 0.243301 | 0.303647 | -0.370184 | -0.267754 | -0.766979 | -0.987687 | 0.024593 | -0.837841 | -0.978592 | -0.449264 |
| **Eccentricity** | 0.267481 | 0.391066 | 0.541972 | 0.019574 | 0.924293 | 1.000000 | 0.269255 | 0.318667 | -0.319362 | -0.297592 | -0.722272 | -0.970313 | 0.019920 | -0.860141 | -0.981058 | -0.449354 |
| **ConvexArea** | 0.999939 | 0.967689 | 0.932607 | 0.951339 | 0.243301 | 0.269255 | 1.000000 | 0.985226 | 0.052564 | -0.206191 | -0.362083 | -0.269922 | -0.847950 | -0.640862 | -0.274024 | -0.362049 |
| **EquivDiameter** | 0.984968 | 0.991380 | 0.961733 | 0.948539 | 0.303647 | 0.318667 | 0.985226 | 1.000000 | 0.028383 | -0.231648 | -0.435945 | -0.327650 | -0.892741 | -0.713069 | -0.330389 | -0.392512 |
| **Extent** | 0.054345 | -0.021160 | -0.078062 | 0.145957 | -0.370184 | -0.319362 | 0.052564 | 0.028383 | 1.000000 | 0.191389 | 0.344411 | 0.354212 | -0.141616 | 0.237956 | 0.347624 | 0.148502 |
| **Solidity** | -0.196585 | -0.303970 | -0.284302 | -0.155831 | -0.267754 | -0.297592 | -0.206191 | -0.231648 | 0.191389 | 1.000000 | 0.607150 | 0.303766 | 0.153388 | 0.343559 | 0.307662 | 0.702163 |
| **roundness** | -0.357530 | -0.547647 | -0.596358 | -0.210344 | -0.766979 | -0.722272 | -0.362083 | -0.435945 | 0.344411 | 0.607150 | 1.000000 | 0.768086 | 0.230273 | 0.782824 | 0.763126 | 0.472149 |
| **Compactness** | -0.268067 | -0.406857 | -0.568377 | -0.015066 | -0.987687 | -0.970313 | -0.269922 | -0.327650 | 0.354212 | 0.303766 | 0.768086 | 1.000000 | -0.009394 | 0.868939 | 0.998686 | 0.484436 |
| **ShapeFactor1** | -0.847958 | -0.864623 | -0.773609 | -0.947204 | 0.024593 | 0.019920 | -0.847950 | -0.892741 | -0.141616 | 0.153388 | 0.230273 | -0.009394 | 1.000000 | 0.469197 | -0.008320 | 0.248619 |
| **ShapeFactor2** | -0.639291 | -0.767592 | -0.859238 | -0.471347 | -0.837841 | -0.860141 | -0.640862 | -0.713069 | 0.237956 | 0.343559 | 0.782824 | 0.868939 | 0.469197 | 1.000000 | 0.872971 | 0.529932 |
| **ShapeFactor3** | -0.272145 | -0.408435 | -0.568185 | -0.019326 | -0.978592 | -0.981058 | -0.274024 | -0.330389 | 0.347624 | 0.307662 | 0.763126 | 0.998686 | -0.008320 | 0.872971 | 1.000000 | 0.484274 |
| **ShapeFactor4** | -0.355721 | -0.429310 | -0.482527 | -0.263749 | -0.449264 | -0.449354 | -0.362049 | -0.392512 | 0.148502 | 0.702163 | 0.472149 | 0.484436 | 0.248619 | 0.529932 | 0.484274 | 1.000000 |

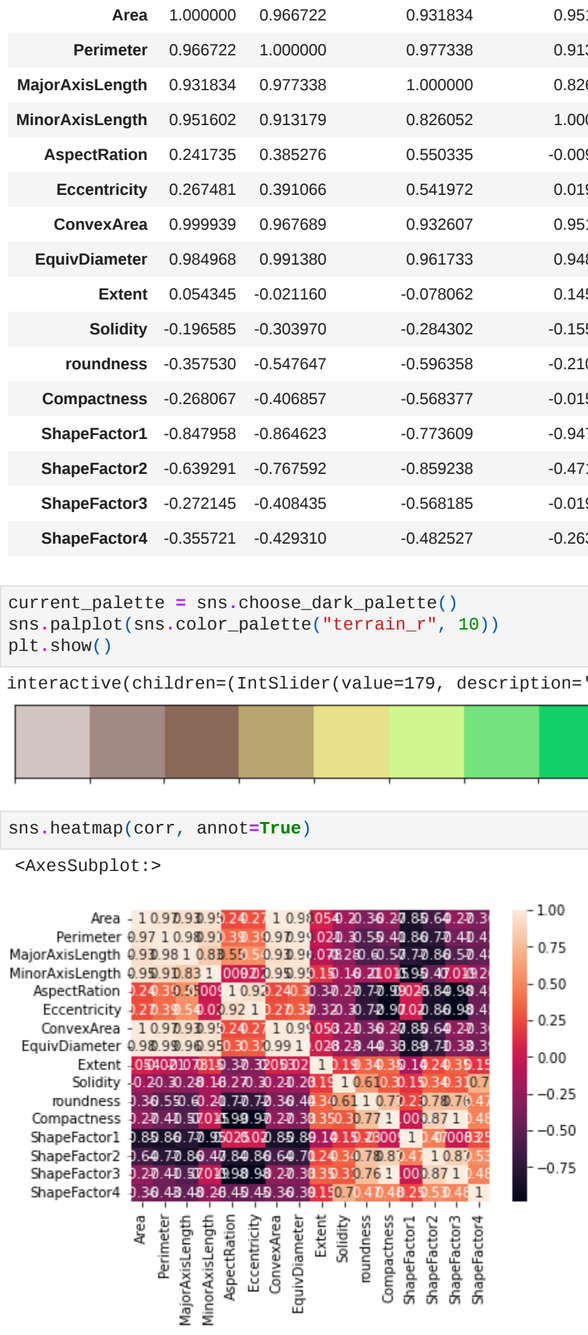
In [259…
```
current_palette = sns.choose_dark_palette()
sns.palplot(sns.color_palette("terrain_r", 10))
plt.show()
```

```
interactive(children=(IntSlider(value=179, description='h', max=359), IntSlider(value=49, description='s', max…
```



In [260…
```
sns.heatmap(corr, annot=True)
```

Out[260]:
```
<AxesSubplot:>
```




In [261…
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [262…
```
X = df.drop('Class', axis=1)
y = df['Class']
```

In [263…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=0)
```

# Random Forest

In [264…
```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators = 100,random_state=35)
clf.fit(X_train, y_train)
```

Out[264]:
```
RandomForestClassifier(random_state=35)
```

In [265…
```
y_pred =clf.predict(X_test)
accuracy = clf.score(X_test, y_test)*100
print('clf Accuracy:', accuracy)
```

```
clf Accuracy: 92.69188395152406
```

# ADA Boost

In [266…
```
from sklearn.ensemble import AdaBoostClassifier
efg = AdaBoostClassifier(n_estimators = 100,random_state=35)
efg.fit(X_train, y_train)
```

Out[266]:
```
AdaBoostClassifier(n_estimators=100, random_state=35)
```

In [247…
```
y_pred =efg.predict(X_test)
accuracy = efg.score(X_test, y_test)*100
print('efg Accuracy:', accuracy)
```

```
efg Accuracy: 65.99338964377525
```

In [ ]:

In [ ]:

In [ ]: