Arya Chakraborty [22MSD7020]

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

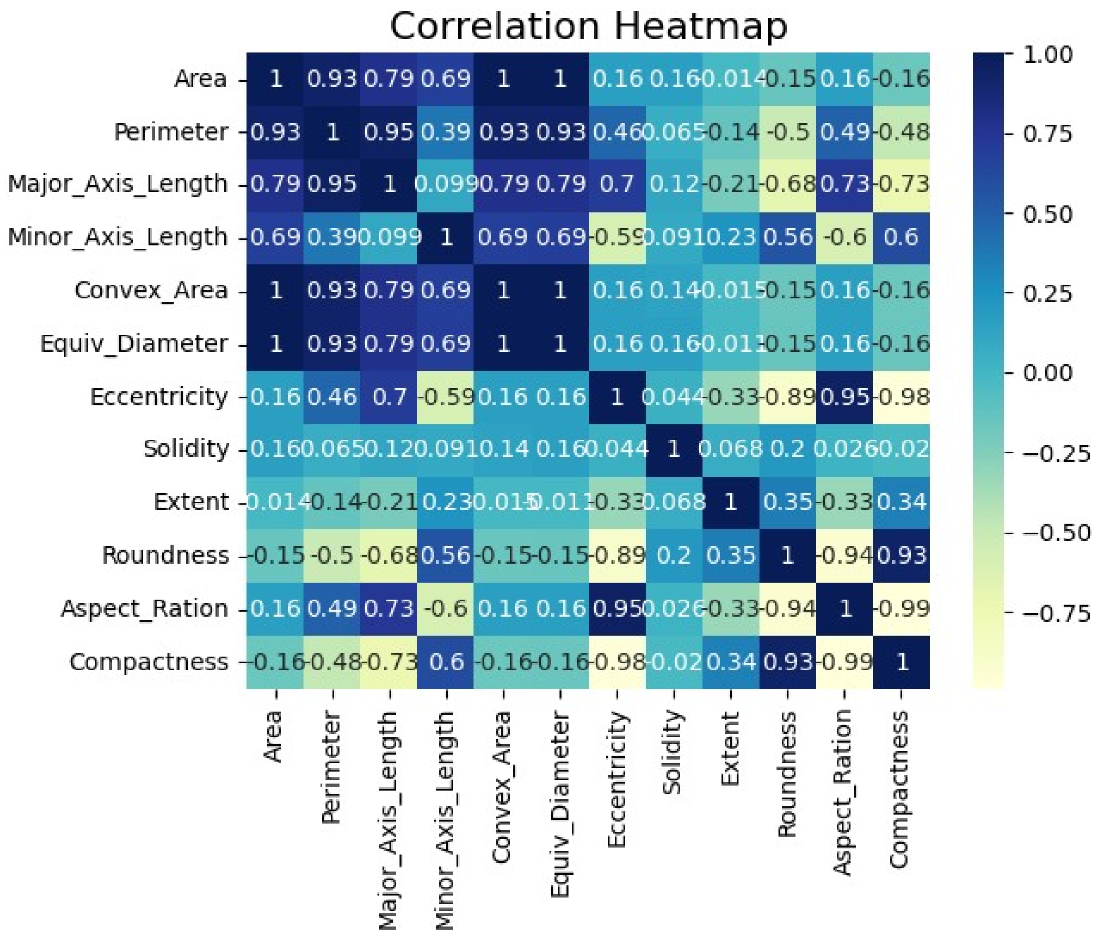
```
seeds = pd.read_excel("Pumpkin_Seeds_Dataset.xlsx")
```

```
seeds.head()
```

```
    Area  Perimeter  Major_Axis_Length  Minor_Axis_Length  Convex_Area
\
0  56276    888.242           326.1485           220.2388        56831

1  76631   1068.146           417.1932           234.2289        77280

2  71623   1082.987           435.8328           211.0457        72663

3  66458    992.051           381.5638           222.5322        67118

4  66107    998.146           383.8883           220.4545        67117


   Equiv_Diameter  Eccentricity  Solidity  Extent  Roundness
Aspect_Ration  \
0        267.6805        0.7376    0.9902  0.7453     0.8963
1.4809
1        312.3614        0.8275    0.9916  0.7151     0.8440
1.7811
2        301.9822        0.8749    0.9857  0.7400     0.7674
2.0651
3        290.8899        0.8123    0.9902  0.7396     0.8486
1.7146
4        290.1207        0.8187    0.9850  0.6752     0.8338
1.7413

   Compactness       Class
0       0.8207  Çerçevelik
1       0.7487  Çerçevelik
2       0.6929  Çerçevelik
3       0.7624  Çerçevelik
4       0.7557  Çerçevelik
```

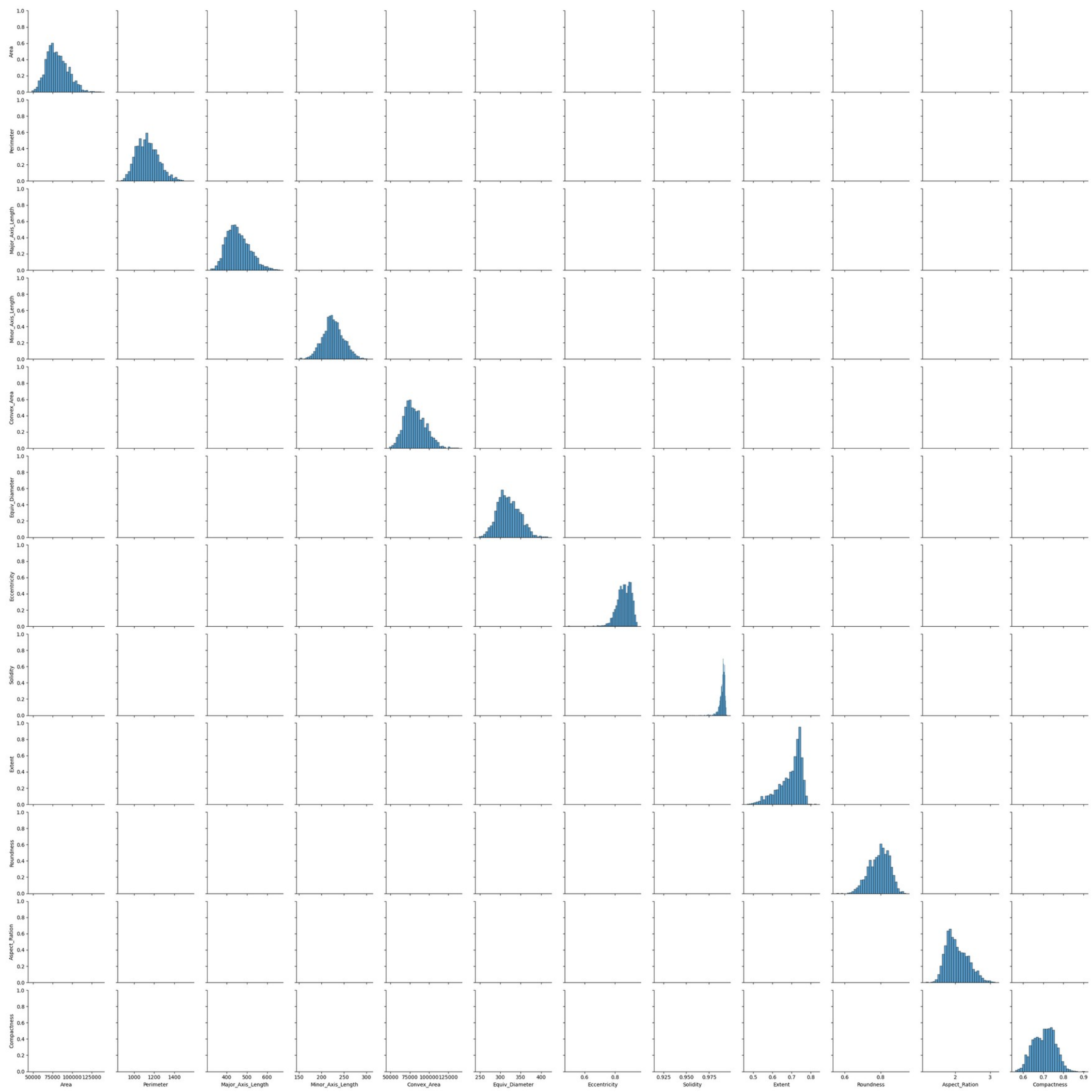
```
sns.heatmap(seeds.corr(), annot=True, cmap="YlGnBu")
plt.title("Correlation Heatmap", fontsize=16)
plt.show()
```

```
C:\Users\chakr\AppData\Local\Temp\ipykernel_13280\3006396044.py:1:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only
valid columns or specify the value of numeric_only to silence this
warning.
  sns.heatmap(seeds.corr(), annot=True, cmap="YlGnBu")
```



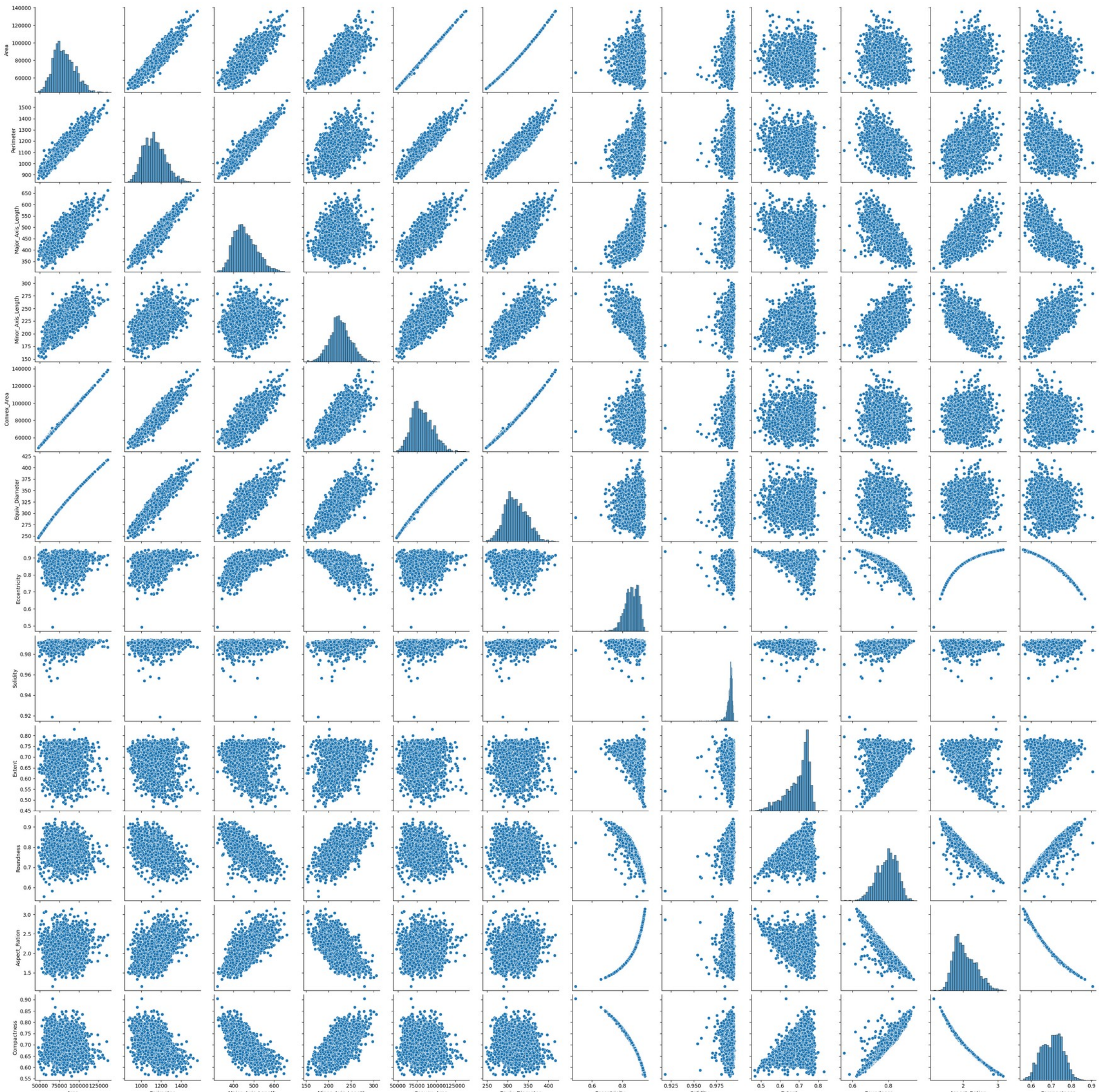

```
sns.pairplot(seeds, kind='box')
```

```
<seaborn.axisgrid.PairGrid at 0x2c2edbd2c50>
```




```
sns.pairplot(seeds)
```

```
<seaborn.axisgrid.PairGrid at 0x2c2ede241f0>
```




```python
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

X = seeds[["Area", "Perimeter",
      "Major_Axis_Length","Minor_Axis_Length", "Convex_Area",
"Equiv_Diameter", "Eccentricity", "Solidity", "Extent", "Roundness",
"Aspect_Ration", "Compactness"]]

y=seeds['Class']

scaler = StandardScaler()
X_scaled=scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled,y,
test_size=0.3, random_state=0)

svm = SVC(kernel='linear')
svm.fit(X_train, y_train)
```

```
SVC(kernel='linear')
```

```python
# Predict the labels of the test set
y_pred = svm.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8533333333333334
```

```python
from sklearn.metrics import classification_report, confusion_matrix

confusion_matrix(y_test,y_pred)
```

```
array([[344,  45],
       [ 65, 296]], dtype=int64)
```

```python
print(classification_report(y_test,y_pred))
```

```
               precision    recall  f1-score   support

   Çerçevelik       0.84      0.88      0.86       389
Ürgüp Sivrisi       0.87      0.82      0.84       361

     accuracy                           0.85       750
    macro avg       0.85      0.85      0.85       750
 weighted avg       0.85      0.85      0.85       750
```

HyperParameter Tuning

```python
from sklearn.model_selection import GridSearchCV
param_grid = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf'], 'gamma':
['scale', 'auto']}

svm1 = SVC()
grid_search = GridSearchCV(svm1, param_grid, cv=5, scoring='accuracy',
n_jobs=-1)

grid_search.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, estimator=SVC(), n_jobs=-1,
             param_grid={'C': [0.1, 1, 10], 'gamma': ['scale',
'auto'],
                         'kernel': ['linear', 'rbf']},
             scoring='accuracy')
```

```python
print("Best parameters:", grid_search.best_params_)
print("Best score:", grid_search.best_score_)
```

```
Best parameters: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}
Best score: 0.8942857142857144
```

```python
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Test accuracy:", accuracy)
```

```
Test accuracy: 0.8626666666666667
```

Conclusion : we can see that during cross validation the score obtained is .89 and the final test score is .86, as the difference is not that big, we can consider that there are no significant overfitting problem present in the model.

To optimize further we can use Bayesian optimization or Random Search CV to see if we can get better result. But considering these best fitted models we can clearly conclude that we have reached the threshold for the parameter tuning.