

# Final Project Unsupervised Machine Learning:

## Dry Bean Dataset



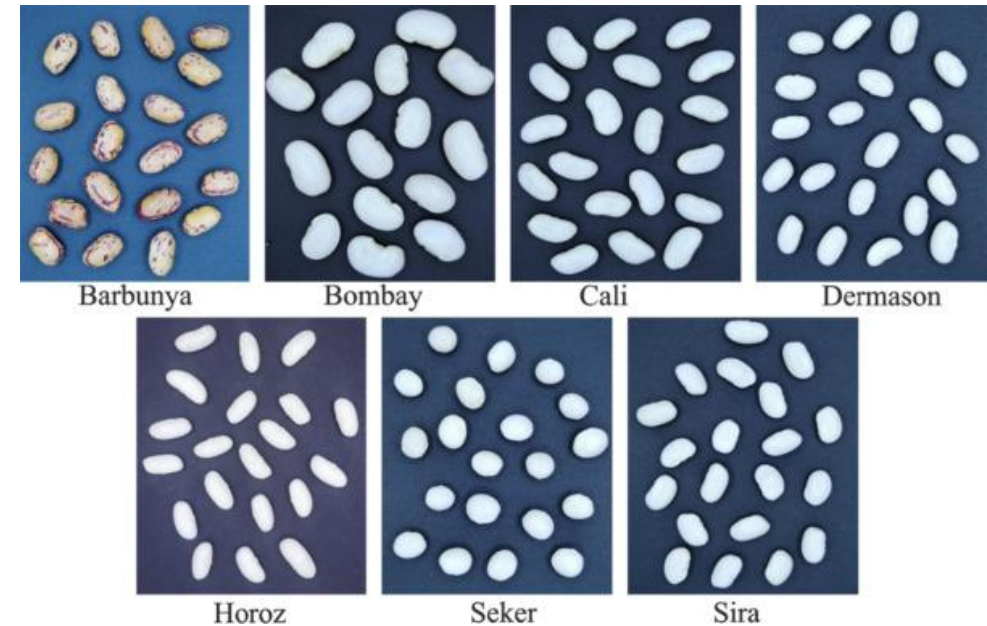
**IBM Machine Learning Professional Certificate**

Course 04: Unsupervised Machine Learning | Dry Bean Dataset

**IBM**

# Contents

- Dataset Description
- Main objectives of the analysis.
- EDA, Data Cleaning, Feature Engineering
- Applying Clustering Algorithms.
- Machine learning analysis and findings.
- Models flaws and advanced steps.



# Data Description Section

# Introduction

ML Algorithms play an essential and promising role in agricultural sector, from this point will discover in this report how we can implement unsupervised learning specifically clustering algorithms to a dataset has Images of 13,611 grains of 7 different registered dry beans were taken with a high-resolution camera. A total of 16 features; 12 dimensions and 4 shape forms, were obtained from the grains.

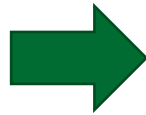
# Dataset Description

## Features Explanations 01

Dataset Dimensions:

❑ (rows, columns) | (13611, 17)

We have 16 features in dry bean dataset in addition of the targeted variable “Class” will explain each of these features in the next slides!



	0	1	2	3	4	5	6	7
Area	28395	28734	29380	30008	30140	30279	30477	30519
Perimeter	610.291	638.018	624.11	645.884	620.134	634.927	670.033	629.727
MajorAxisLength	208.178	200.525	212.826	210.558	201.848	212.561	211.05	212.997
MinorAxisLength	173.889	182.734	175.931	182.517	190.279	181.51	184.039	182.737
AspectRatio	1.19719	1.09736	1.20971	1.15364	1.0608	1.17107	1.14677	1.16559
Eccentricity	0.549812	0.411785	0.562727	0.498616	0.33368	0.520401	0.489478	0.51376
ConvexArea	28715	29172	29690	30724	30417	30600	30970	30847
EquivDiameter	190.141	191.273	193.411	195.467	195.897	196.348	196.989	197.124
Extent	0.763923	0.783968	0.778113	0.782681	0.773098	0.775688	0.762402	0.770682
Solidity	0.988856	0.984986	0.989559	0.976696	0.990893	0.98951	0.984081	0.989367
roundness	0.958027	0.887034	0.947849	0.903936	0.984877	0.943852	0.85308	0.967109
Compactness	0.913358	0.953861	0.908774	0.928329	0.970516	0.923726	0.933374	0.92548
ShapeFactor1	0.00733151	0.00697866	0.00724391	0.00701673	0.00669701	0.00702007	0.0069249	0.00697915
ShapeFactor2	0.00314729	0.00356362	0.00304773	0.00321456	0.00366497	0.00315278	0.00324202	0.00315829
ShapeFactor3	0.834222	0.909851	0.825871	0.861794	0.9419	0.85327	0.871186	0.856514
ShapeFactor4	0.998724	0.99843	0.999066	0.994199	0.999166	0.999236	0.999049	0.998345
Class	SEKER	SEKER	SEKER	SEKER	SEKER	SEKER	SEKER	SEKER

# Dataset Description

## Features Explanations 02

1.) **Area (A)**: The area of a bean zone and the number of pixels within its boundaries.

2.) **Perimeter (P)**: Bean circumference is defined as the length of its border.

3.) **Major axis length (L)**: The distance between the ends of the longest line that can be drawn from a bean.

4.) **Minor axis length (l)**: The longest line that can be drawn from the bean while standing perpendicular to the main axis.

5.) **Aspect ratio (K)**: Defines the relationship between L and l.

6.) **Eccentricity (Ec)**: Eccentricity of the ellipse having the same moments as the region.

7.) **Convex area (C)**: Number of pixels in the smallest convex polygon that can contain the area of a bean seed.

8.) **Equivalent diameter (Ed)**: The diameter of a circle having the same area as a bean seed area.

9.) **Extent (Ex)**: The ratio of the pixels in the bounding box to the bean area.

10.) **Solidity (S)**: Also known as convexity. The ratio of the pixels in the convex shell to those found in beans.

# Dataset Description

## Features Explanations 03

12.) Compactness (CO): Measures the roundness of an object: Ed/L

13.) ShapeFactor1 (SF1)

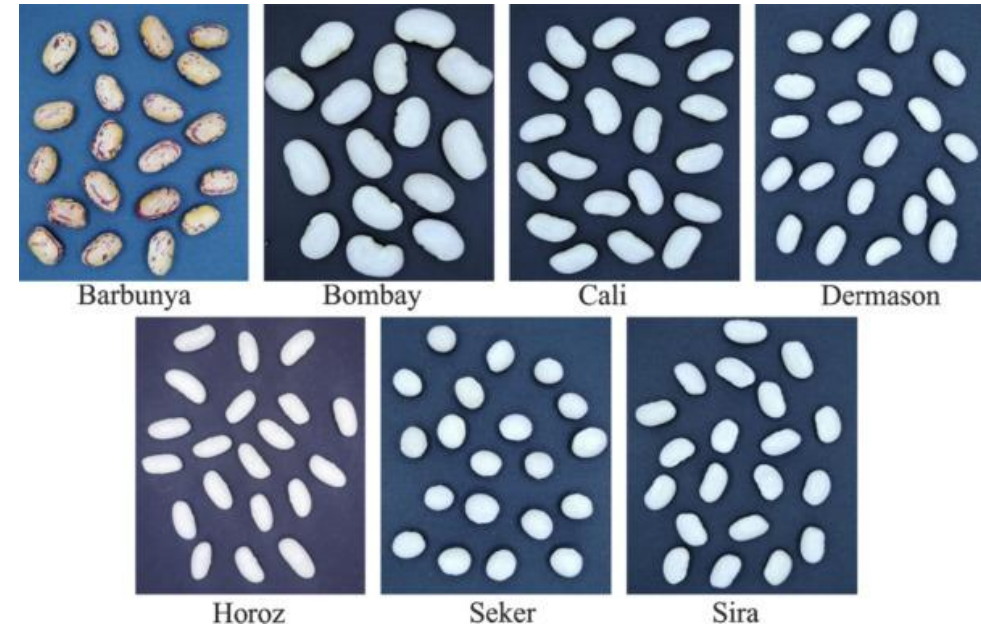
14.) ShapeFactor2 (SF2)

15.) ShapeFactor3 (SF3)

16.) ShapeFactor4 (SF4)

17.) Class: (Barbunya, Bombay, Cali, Dermosan, Horoz, Seker and Sira)

We have 7 different classes of beans. So, when we apply clustering methods, we expect to find the same number of Clusters! using clustering algorithms.



# Dataset Description

## Features statistical description

	count	mean	std	min	25%	50%	75%	max
Area	13611.00	53048.28	29324.10	20420.00	36328.00	44652.00	61332.00	254616.00
Perimeter	13611.00	855.28	214.29	524.74	703.52	794.94	977.21	1985.37
MajorAxisLength	13611.00	320.14	85.69	183.60	253.30	296.88	376.50	738.86
MinorAxisLength	13611.00	202.27	44.97	122.51	175.85	192.43	217.03	460.20
AspectRatio	13611.00	1.58	0.25	1.02	1.43	1.55	1.71	2.43
Eccentricity	13611.00	0.75	0.09	0.22	0.72	0.76	0.81	0.91
ConvexArea	13611.00	53768.20	29774.92	20684.00	36714.50	45178.00	62294.00	263261.00
EquivDiameter	13611.00	253.06	59.18	161.24	215.07	238.44	279.45	569.37
Extent	13611.00	0.75	0.05	0.56	0.72	0.76	0.79	0.87
Solidity	13611.00	0.99	0.00	0.92	0.99	0.99	0.99	0.99
roundness	13611.00	0.87	0.06	0.49	0.83	0.88	0.92	0.99
Compactness	13611.00	0.80	0.06	0.64	0.76	0.80	0.83	0.99
ShapeFactor1	13611.00	0.01	0.00	0.00	0.01	0.01	0.01	0.01
ShapeFactor2	13611.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ShapeFactor3	13611.00	0.64	0.10	0.41	0.58	0.64	0.70	0.97
ShapeFactor4	13611.00	1.00	0.00	0.95	0.99	1.00	1.00	1.00





# Main Objective of the analysis:

In this section we will explore the dataset in-depth through several EDA techniques such as checking null values, data skewness, and data visualization, furthermore, showing the correlation between the features for the sake of feature engineering implementation and data cleaning.

# Exploratory Data Analysis (EDA) + Feature Engineering Section

# Exploratory Data Analysis

## Checking for Null values

```
Area          0
Perimeter     0
MajorAxisLength  0
MinorAxisLength  0
AspectRatio    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
```



Great, **no missing values** within  
our features !

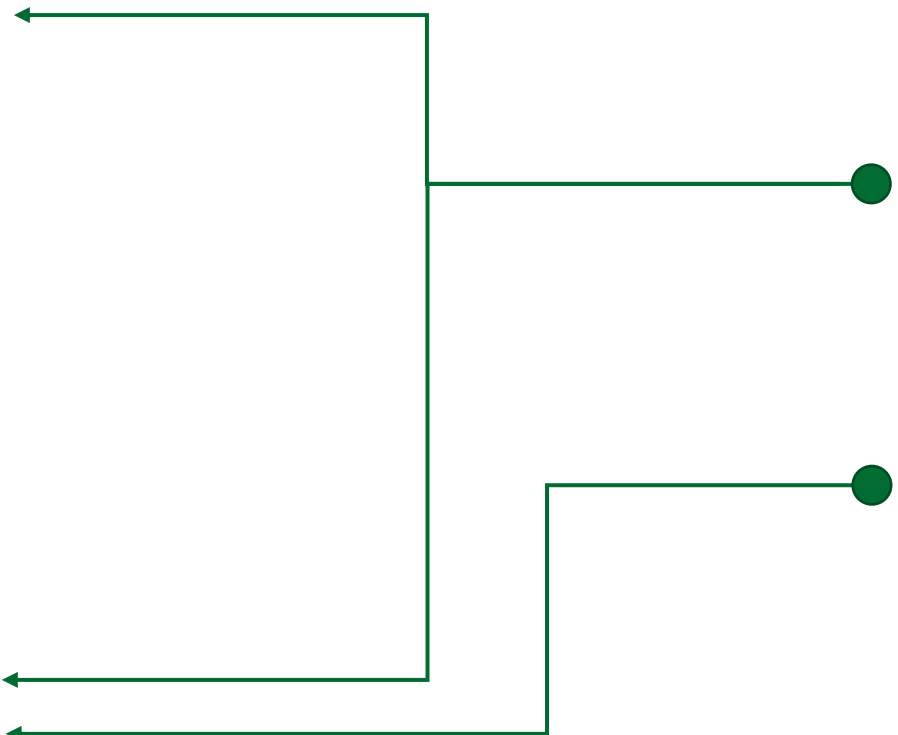
# Exploratory Data Analysis

## - Identifying categorical features and numerical features:

Area	int64		
Perimeter	float64		
MajorAxisLength	float64		
MinorAxisLength	float64		
AspectRatio	float64		
Eccentricity	float64		
ConvexArea	int64		
EquivDiameter	float64		
Extent	float64		
Solidity	float64		
roundness	float64		
Compactness	float64		
ShapeFactor1	float64		
ShapeFactor2	float64		
ShapeFactor3	float64		
ShapeFactor4	float64		
Class	object		
dtype: object			

Integers + Float  
(Numerical Features)

Categorical Features

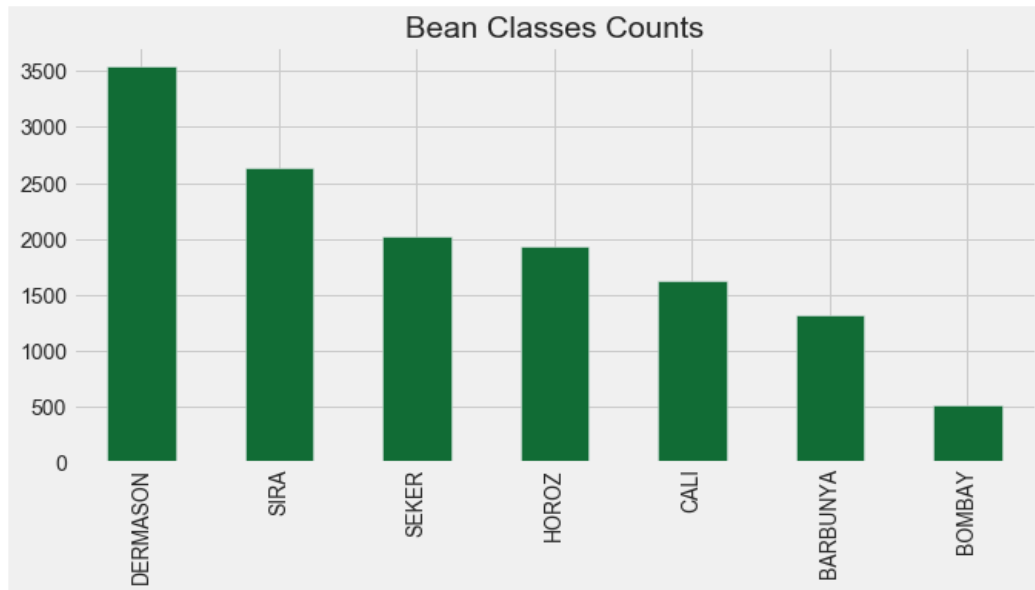


# Exploratory Data Analysis

## Exploring the target variable :

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x21a02051940>
```

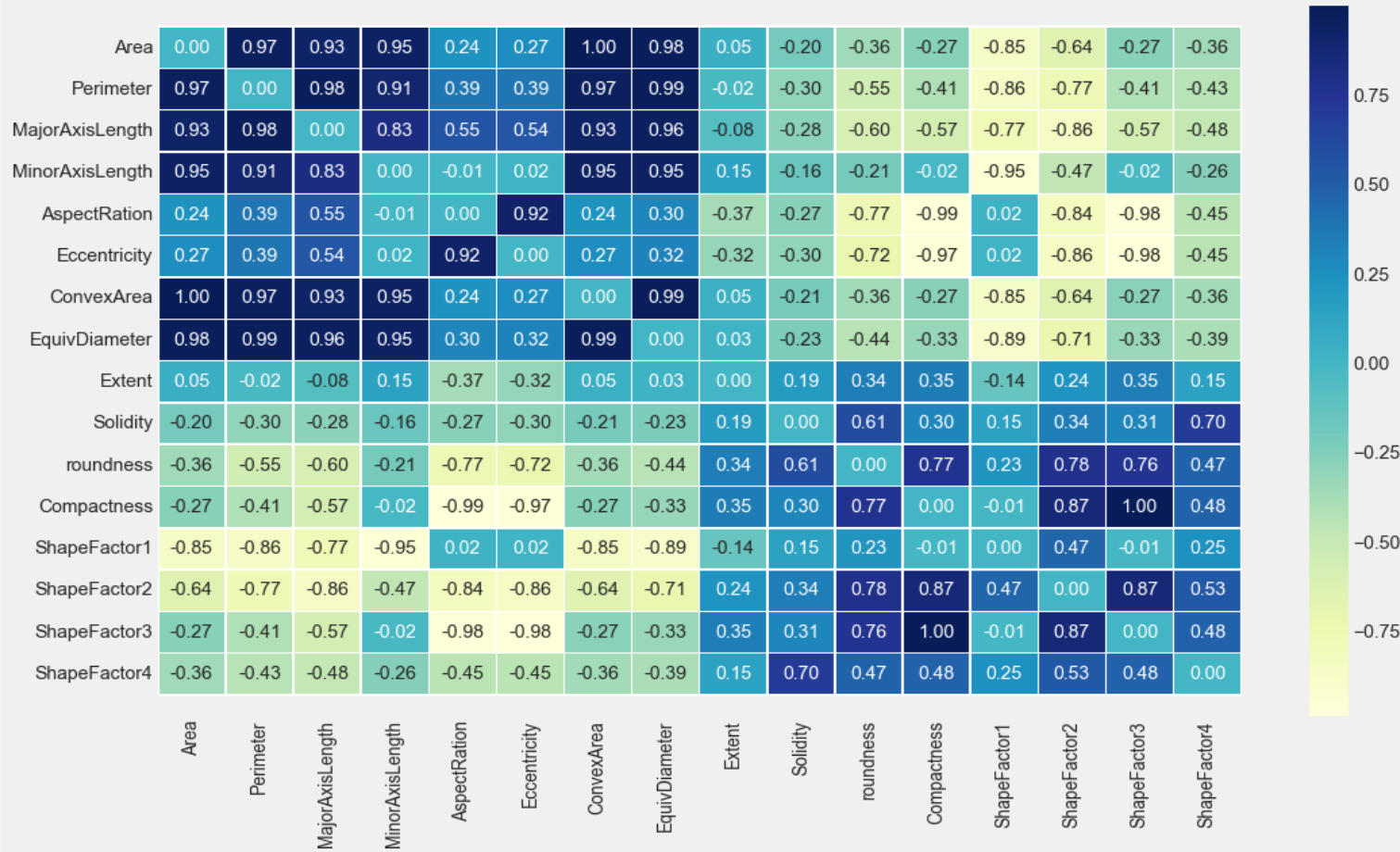


As shown, we have unbalanced bean dataset, since there are less observations for some classes as BARBUNYA and BOMBAY, in this presentation we will discover if this could affect the clustering process.

	Class%
DERMASON	26.05
SIRA	19.37
SEKER	14.89
HOROZ	14.17
CALI	11.98
BARBUNYA	9.71
BOMBAY	3.84

# Exploratory Data Analysis

## - Studying the correlations between features using Heat Map!



The goal of this matrix is to show the **correlations** between features, and this is useful for feature engineering techniques in the coming slides will show these **correlations** more clearly through data frames.

# Exploratory Data Analysis

## - Studying the highest correlations between the features

	Feature_one	Feature_two	correlation
0	Area	ConvexArea	1.00
1	ConvexArea	Area	1.00
2	Compactness	ShapeFactor3	1.00
3	ShapeFactor3	Compactness	1.00
4	Perimeter	EquivDiameter	0.99
5	EquivDiameter	Perimeter	0.99
6	AspectRatio	Compactness	0.99
7	Eccentricity	ShapeFactor3	0.98
8	MajorAxisLength	Perimeter	0.98
9	MinorAxisLength	Area	0.95
10	ShapeFactor1	MinorAxisLength	0.95
11	ShapeFactor2	ShapeFactor3	0.87
12	roundness	ShapeFactor2	0.78
13	Solidity	ShapeFactor4	0.70
14	ShapeFactor4	Solidity	0.70
15	Extent	AspectRatio	0.37

We can notice that there is high correlation between the features because most of them are geometric features describe the shape of the bean.





# Exploratory Data Analysis

## Skewness examination in the dataset

Skewness examination in the features in anticipation of transformations.

We take in our consideration the following:

- (nearly 0) or  $(-0.75 < \text{value} < 0.75)$  no skewness
- Positive (value  $> +0.75$ ): right skewness
- negative (value  $< -0.75$ ): left skewness

### All features

skewness_value	
Area	2.95
ConvexArea	2.94
MinorAxisLength	2.24
EquivDiameter	1.95
Perimeter	1.63
MajorAxisLength	1.36
AspectRatio	0.58
ShapeFactor2	0.30
ShapeFactor3	0.24
Compactness	0.04
ShapeFactor1	-0.53
roundness	-0.64
Extent	-0.90
Eccentricity	-1.06
Solidity	-2.55
ShapeFactor4	-2.76

### Feature should be transformed

skewness_value	
Area	2.95
ConvexArea	2.94
MinorAxisLength	2.24
EquivDiameter	1.95
Perimeter	1.63
MajorAxisLength	1.36
Extent	-0.90
Eccentricity	-1.06
Solidity	-2.55
ShapeFactor4	-2.76

# Feature Engineering

## Applying Log transformation to right skewed data.

Note: when we apply log transformation on the features, only positive skewness will be handled since log transformation tries to approach right skewness into left skewness to reach out the symmetric or normal distribution.

	skewness_value
Area	2.95
ConvexArea	2.94
MinorAxisLength	2.24
EquivDiameter	1.95
Perimeter	1.63
MajorAxisLength	1.36

Log  
Transformation

	skewness_value
MinorAxisLength	1.31
EquivDiameter	1.07
Area	1.07
ConvexArea	1.07
Perimeter	0.84
MajorAxisLength	0.63
AspectRatio	0.58
ShapeFactor2	0.30
ShapeFactor3	0.24
Compactness	0.04
ShapeFactor1	-0.53
roundness	-0.64
Extent	-0.90
Eccentricity	-1.06
Solidity	-2.55
ShapeFactor4	-2.76

# Feature Engineering

## Features Scaling

Feature scaling is relevant in machine learning models that compute some sort of distance metric, where we are going to use clustering methods like K-means which depends on distance metric we must scale our features to perform clustering in the appropriate way!

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRatio	Eccentricity	ConvexArea	EquivDiameter	Extent	Solidity	roundness	Compactness	ShapeFactor1	ShapeFactor2	ShapeFactor3
0	10.25	6.42	5.34	5.16	1.20	0.55	10.27	5.25	0.76	0.99	0.96	0.91	0.01	0.00	0.83
1	10.27	6.46	5.31	5.21	1.10	0.41	10.28	5.26	0.78	0.98	0.89	0.95	0.01	0.00	0.91
2	10.29	6.44	5.37	5.18	1.21	0.56	10.30	5.27	0.78	0.99	0.95	0.91	0.01	0.00	0.83
3	10.31	6.47	5.35	5.21	1.15	0.50	10.33	5.28	0.78	0.98	0.90	0.93	0.01	0.00	0.86
4	10.31	6.43	5.31	5.25	1.06	0.33	10.32	5.28	0.77	0.99	0.98	0.97	0.01	0.00	0.94
5	10.32	6.46	5.36	5.21	1.17	0.52	10.33	5.28	0.78	0.99	0.94	0.92	0.01	0.00	0.85
6	10.32	6.51	5.36	5.22	1.15	0.49	10.34	5.29	0.76	0.98	0.85	0.93	0.01	0.00	0.87



	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRatio	Eccentricity	ConvexArea	EquivDiameter	Extent	Solidity	roundness	Compactness	ShapeFactor1	ShapeFactor2	ShapeFactor3
0	-1.28	-1.38	-1.63	-0.68	-1.57	-2.19	-1.28	-1.28	0.29	0.37	1.42	1.84	0.68	2.40	1.93
1	-1.25	-1.19	-1.78	-0.43	-1.97	-3.69	-1.24	-1.25	0.70	-0.46	0.23	2.50	0.37	3.10	2.69
2	-1.20	-1.28	-1.54	-0.62	-1.51	-2.05	-1.20	-1.20	0.58	0.52	1.25	1.76	0.60	2.24	1.84
3	-1.15	-1.13	-1.58	-0.43	-1.74	-2.74	-1.12	-1.15	0.67	-2.24	0.52	2.08	0.40	2.52	2.20
4	-1.14	-1.31	-1.75	-0.21	-2.12	-4.54	-1.14	-1.14	0.48	0.80	1.87	2.77	0.12	3.27	3.01
5	-1.12	-1.21	-1.54	-0.46	-1.67	-2.51	-1.13	-1.12	0.53	0.51	1.19	2.01	0.40	2.41	2.12
6	-1.11	-0.97	-1.57	-0.39	-1.77	-2.84	-1.10	-1.11	0.26	-0.66	-0.34	2.16	0.32	2.56	2.30

# Machine Learning Analysis & Findings

# Machine Learning Analysis & Findings

In the following slides we will compare between 4 different Clustering methods k-means, Agglomerative Hierarchical clustering, DBSCAN, MeanShift in terms of finding the appropriate number of clusters and comparing the clustered observations with the actual classes in the dry bean dataset.

# Machine Learning Analysis

## K-means Algorithm 01

```
### BEGIN SOLUTION
# Create and fit a range of models
km_list = list()

for clust in range(1,21):
    km = KMeans(n_clusters=clust, random_state=42)
    km = km.fit(df[feature_columns])

    km_list.append(pd.Series({'clusters': clust,
                              'inertia': km.inertia_,
                              'model': km}))

plot_data = (pd.concat(km_list, axis=1)
              .T
              [['clusters', 'inertia']]
              .set_index('clusters'))

ax = plot_data.plot(marker='o', ls='-')
ax.set_xticks(range(0,21,2))
ax.set_xlim(0,21)
ax.set_xlabel('Cluster', ylabel='Inertia');
### END SOLUTION
```

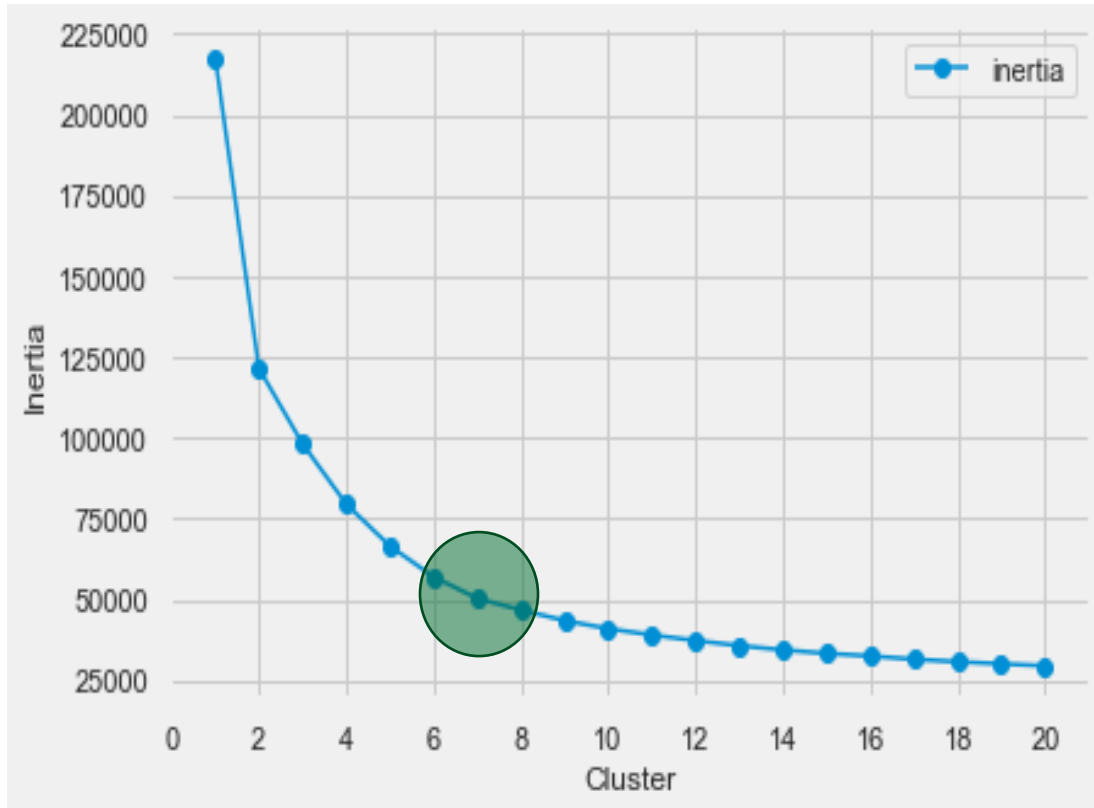
Here we have Implemented K-means algorithm, with a range of different K values (0 – 20) for the sake of finding the appropriate number of clusters in dry bean dataset, and to measure the entropy in the model we've selected the inertia metric and elbow method to find the appropriate value of K (no. clusters)

**Inertia** : is defined as the sum of squared distance from each point ( $X_i$ ) to its cluster  $C_k$

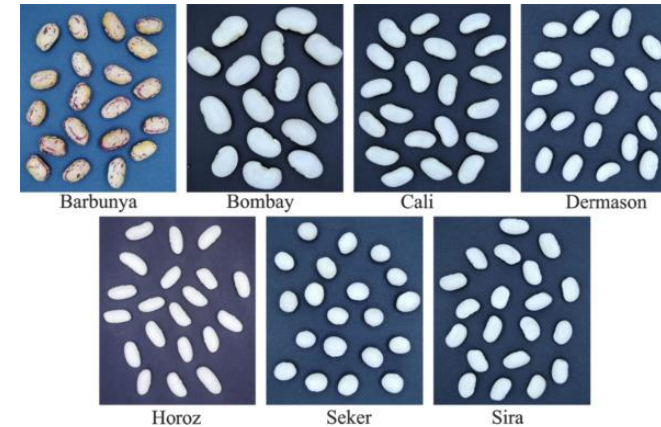
$$\sum_{i=1}^n (X_i - C_k)^2$$

# Machine Learning Analysis

## K-means Algorithm 02



As shown in the graph, in case we follow elbow method approach we can clearly obtain that the appropriate number of clusters ranges between 6 – 8 where, this is aligned with our dataset which contains seven different clusters (Classes)



# Machine Learning Analysis

## K-means Algorithm 03

Applying K-means with no. clusters = 7

```
### BEGIN SOLUTION
km = KMeans(n_clusters=7, random_state=42)
km = km.fit(df[float_columns])
df['k-means'] = km.predict(df[float_columns])
df.sample(7)
```

Class Label	Cluster Number
BARBUNYA	0
DERMASON	1
HORZO	2
Cali	3
BOMBAY	4
SIRA	5
SEKER	6

After applying K-means algorithm with no. clusters = 7 to the dry bean dataset, it will classify each observation to the 7 clusters which we assigned in the model then we will see how many of these observations are classified correctly.

ShapeFactor1	ShapeFactor2	ShapeFactor3	ShapeFactor4	Class	kmeans
0.02	1.60	1.69	0.98	SEKER	6
1.54	0.17	-0.48	0.39	DERMASON	1
-0.13	-1.01	-1.14	0.80	BARBUNYA	2
-0.92	-0.84	-0.47	0.04	BARBUNYA	0
-0.06	1.66	1.80	0.64	SEKER	6
-1.30	-0.18	0.72	0.85	BARBUNYA	0
1.91	1.08	0.19	0.93	DERMASON	1
0.46	-1.12	-1.56	-1.76	HORZO	2
1.37	1.47	0.77	0.73	DERMASON	1
-1.60	-0.76	0.12	-0.13	BARBUNYA	0



# Machine Learning Analysis

## K-means Algorithm 04

Here we group by Class and k-means columns to validate our clustering model with the actual classes labeling:

		number
Class	k-means	
BARBUNYA	0	1176
	2	8
	3	40
	4	1
	5	84
BOMBAY	6	13
	3	2
	4	520
CALI	0	1321
	2	24
	3	268
	4	2
	5	13
	6	2

DERMASON	1	2735
	2	5
	3	7
	5	685
	6	114
HORZO	0	30
	1	4
	2	1660
	3	188
	5	46
SEKER	1	14
	3	3
	5	133
SIRA	6	1877
	0	13
	1	71
	2	83
	3	25
	5	2422
	6	22



Cluster index	Class Label	Correct : all	Correct Clustering percentage
0	BARBUNYA	1176 : 1322	88.96 %
1	DERMASON	2735 : 3546	77.13 %
2	HORZO	1660 : 1928	86.10 %
3	CALI	268 : 1630	16.44 %
4	BOMBAY	520 : 522	99.17 %
5	SIRA	2422 : 2636	91.88 %
6	SEKER	1877 : 2027	92.60 %

The outcomes were very good and satisfactory for 5 clusters, , but only one of the clusters (**CALI**) achieved poor results because of characteristics similarity with another cluster (**BARBUNYA**)

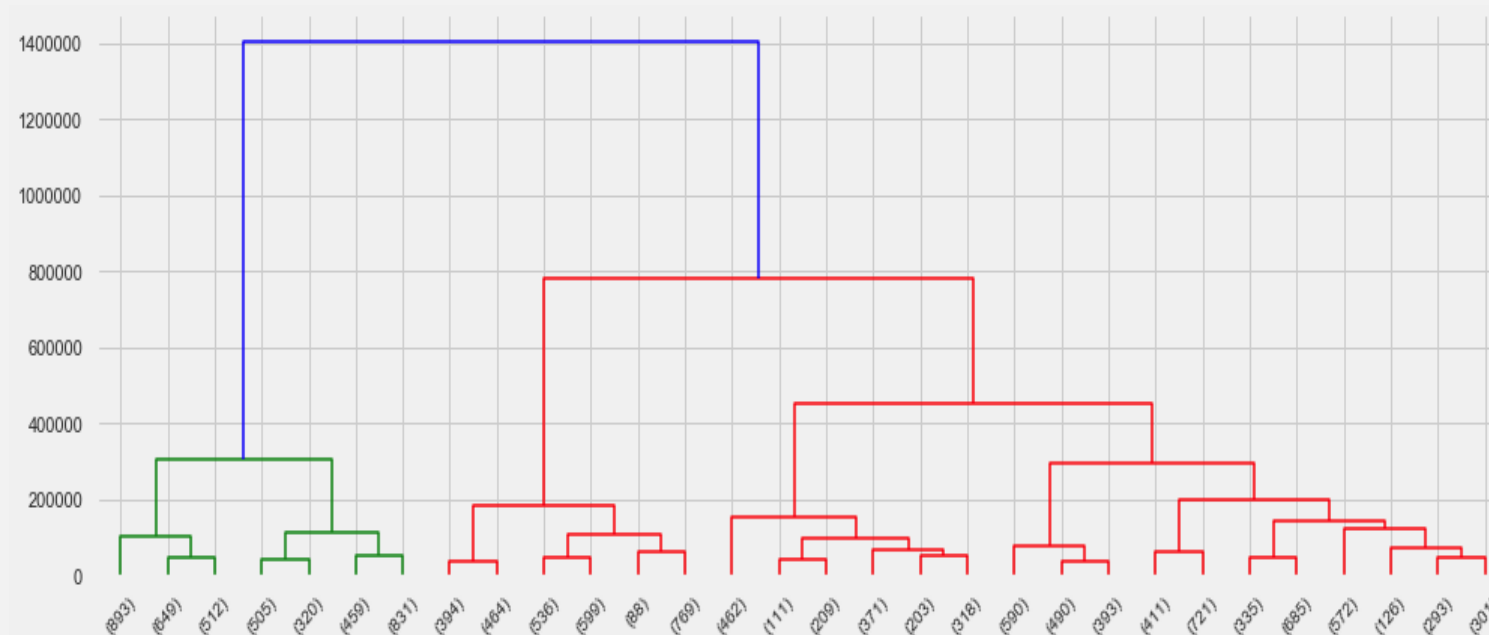
# Machine Learning Analysis

## Agglomerative Clustering Algorithm 01

Applying agglomerative hierarchical clustering  
with no. clusters = 7

```
from sklearn.cluster import AgglomerativeClustering
### BEGIN SOLUTION
ag = AgglomerativeClustering(n_clusters=7, linkage='ward', compute_full_tree=True)
ag = ag.fit(df[float_columns])
df['agglom'] = ag.fit_predict(df[float_columns])
```

Class Label	Cluster Number
BARBUNYA	0
HORZO	1
SIRA	2
SEKER	3
BOMBAY	4
DERMASON	5
CALI	6



# Machine Learning Analysis

## Agglomerative Clustering Algorithm 02

Here we group by Class and agglom columns to validate our clustering model with the actual classes labeling:

number		
Class	agglom	
BARBUNYA	0	1273
	1	2
	2	32
	3	6
	5	8
	6	1
BOMBAY	4	522
CALI	0	1572
	1	32
	2	18
	3	2
	6	6

DERMASON	0	1
	1	1
	2	494
	3	80
	5	2969
	6	1
HOROZ	0	52
	1	1602
	2	167
	5	12
	6	95
SEKER	0	7
	2	60
	3	1879
	5	80
	6	1
SIRA	0	22
	1	33
	2	2252
	3	57
	5	266
	6	6



Cluster index	Class Label	Correct : all	Correct Clustering percentage
0	BARBUNYA	1273 : 1322	96.30 %
5	DERMASON	2969 : 3546	83.73 %
1	HORZO	1602 : 1928	83.09 %
6	CALI	6 : 1630	0.37 %
4	BOMBAY	522 : 522	100 %
2	SIRA	2252 : 2636	85.43 %
3	SEKER	1879 : 2027	92.70 %

Again (CALI) Cluster achieved very poor results because of characteristics similarity (BARBUNYA) cluster

# Machine Learning Analysis

## DBSCAN and MeanShift Algorithms

```
from sklearn.cluster import DBSCAN
dbs = DBSCAN(eps=0.5, min_samples=11, metric='euclidean')
dbs = dbs.fit(df[float_columns])
```

```
from sklearn.cluster import MeanShift
ms = MeanShift(bandwidth=2.8, n_jobs=-1)
ms = ms.fit(df[float_columns])
```

```
array([-1,  0,  1,  2,  3,  4,  5,  6], dtype=int64)
```

After applying these two algorithms many times with different parameters we got poor outcomes, where we validated the clustered observations with the actual classes, we got high error though the algorithm predicted the appropriate number of clusters

		number	
Class	dbscan		
BARBUNYA	-1	1319	
	0	3	
BOMBAY	-1	522	
	-1	1589	
CALI	0	4	
	1	37	
	-1	1264	
DERMASON	0	2277	
	6	5	

		number	
Class	MeanShift		
BARBUNYA	0	974	
	1	346	
	2	1	
BOMBAY	4	1	
	1	508	
	2	14	
CALI	0	891	
	1	710	
	2	29	

# Machine Learning Findings

## algorithms comparison

As we mentioned in the previous slide DBSCAN and MeanShift achieved poor results in terms of observations clustering so will be excluded from the comparison and we will compare only between **K-means** and **Agglomerative Clustering Algorithm**

Class Label	Correct Clustering Percentage K-means	Correct Clustering Percentage Agglomerative
BARBUNYA	88.96 %	96.30 %
DERMASON	77.13 %	83.73 %
HORZO	86.10 %	83.09 %
CALI	16.44 %	0.37 %
BOMBAY	99.17 %	100 %
SIRA	91.88 %	85.43 %
SEKER	92.60 %	92.70 %
Overall Accuracy	92.05 %	90.03 %

Both clustering methods achieved great Overall accuracy but at the end will choose the best model one which is **K-means** with overall accuracy = 92.05%.

# Models flaws and strengths and advanced steps

# Models flaws and strengths

## Models Flaws and Strength:

K-means and agglomerative hierarchical clustering methods both were fast and accurate in terms of finding the appropriate number of clusters in addition of that they clustered most majority of bean observations correctly except one class which was **CALI** due to its characteristic's similarity with **BARBUNYA** class but in overall both algorithms achieved above 90 % in terms of clustering accuracy.

In contrast, DBSCAN and MeanShift required a lot of time to find the appropriate number of clusters since they identify the number of clusters dependably on the parameters that we need to change again and again to find the expected number of clusters furthermore, the outcomes of the observations clustering were so poor.

# Advanced steps

## further suggestions:

- To enhance the detection of **CALI** cluster in K-means and AHC models we can add another column represents the color of the bean to avoid the similarity issue with **BARBUNYA**



Cali



Barbunya

- To enhance the clustering process in DBSCAN and MeanShift we can use bean images as data instead of geometric characteristics of bean because both algorithms play very good roles in computer vision applications or we can use grid search and hyperparameters tuning to find the best parameters, but this will require a lot of time especially with MeanShift model





# Thank you

## IBM Machine Learning Professional Certificate

Unsupervised Machine Learning

By: Mohamad Osman