**Clustering Code Along**

We'll be working with a real data set about seeds, from UCI repository:

<https://archive.ics.uci.edu/ml/datasets/seeds>.

The examined group comprised kernels belonging to three different varieties of wheat: Kama, Rosa and Canadian, 70 elements each, randomly selected for the experiment. High quality visualization of the internal kernel structure was detected using a soft X-ray technique. It is non-destructive and considerably cheaper than other more sophisticated imaging techniques like scanning microscopy or laser technology. The images were recorded on 13x18 cm X-ray KODAK plates. Studies were conducted using combine harvested wheat grain originating from experimental fields, explored at the Institute of Agrophysics of the Polish Academy of Sciences in Lublin.

The data set can be used for the tasks of classification and cluster analysis.

Attribute Information:

To construct the data, seven geometric parameters of wheat kernels were measured:

1. area A,
2. perimeter P,
3. compactness C = 4*pi*A/P^2,
4. length of kernel,
5. width of kernel,
6. asymmetry coefficient
7. length of kernel groove. All of these parameters were real-valued continuous.

Let's see if we can cluster them in to 3 groups with K-means!

**from** **pyspark.sql** **import** SparkSession spark = SparkSession.builder.appName('cluster').getOrCreate()

**from** **pyspark.ml.clustering** **import** KMeans

*# Loads data.*

dataset = spark.read.csv("seeds\_dataset.csv",header=**True**,inferSchema=**True**)

dataset.head()

dataset.describe().show()

**Format the Data**

**from** **pyspark.ml.linalg** **import** Vectors

**from** **pyspark.ml.feature** **import** VectorAssembler

dataset.columns

['area', 'perimeter', 'compactness', 'length\_of\_kernel', 'width\_of\_kernel', 'asymmetry\_coefficient', 'length\_of\_groove']

vec\_assembler = VectorAssembler(inputCols = dataset.columns, outputCol='features')

final\_data = vec\_assembler.transform(dataset)

## Scale the Data

It is a good idea to scale our data to deal with the curse of dimensionality

**from** **pyspark.ml.feature** **import** StandardScaler

scaler = StandardScaler(inputCol="features", outputCol="scaledFeatures", withStd=**True**, withMean=**False**)

*# Compute summary statistics by fitting the StandardScaler*

scalerModel = scaler.fit(final\_data)

*# Normalize each feature to have unit standard deviation.*

final\_data = scalerModel.transform(final\_data)

**Train the Model and Evaluate**

*# Trains a k-means model.* kmeans = KMeans(featuresCol='scaledFeatures',k=3) model = kmeans.fit(final\_data)

*# Evaluate clustering by computing Within Set Sum of Squared Errors.*

wssse = model.computeCost(final\_data)

print("Within Set Sum of Squared Errors = " + str(wssse))

*# Shows the result.*

centers = model.clusterCenters()

print("Cluster Centers: ")

**for** center **in** centers:

print(center)

model.transform(final\_data).select('prediction').show()