Machine Learning: Unsupervised Learning - KMeans

Import Python Libraries and Modules

```
# Import Python Libraries: NumPy and Pandas
import pandas as pd
import numpy as np

# Import Libraries & modules for data visualization
from pandas.plotting import scatter_matrix
from matplotlib import pyplot


# Import scikit-Learn module for the algorithm/modeL: K-Means
from sklearn.cluster import KMeans
```

Load data-set (iris.csv)

```
# Specify location of the dataset
filename = 'iris.csv'

# Load the data into a Pandas DataFrame
df = pd.read_csv(filename)
```

Pre-process Dataset (Clean Data)

```
# mark zero values as missing or NaN
df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']] \
= df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm'
]].replace(0,np.NaN)

# count the number of NaN values in each column
print(df.isnull().sum())
```

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
```

```
Species          0
dtype: int64
```

Perform the exploratory data analysis (EDA) on the dataset

```
# to get the dimension of the dataset, no. of rows x no. of columns
print(df.shape)
```

```
(150, 6)
```

```
# to get data types of all variables
print(df.dtypes)
```

```
Id                 int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

```
#display first 5 rows
print(df.head(5))
```

```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
```

```
#display summary statistics
print(df.describe())
```

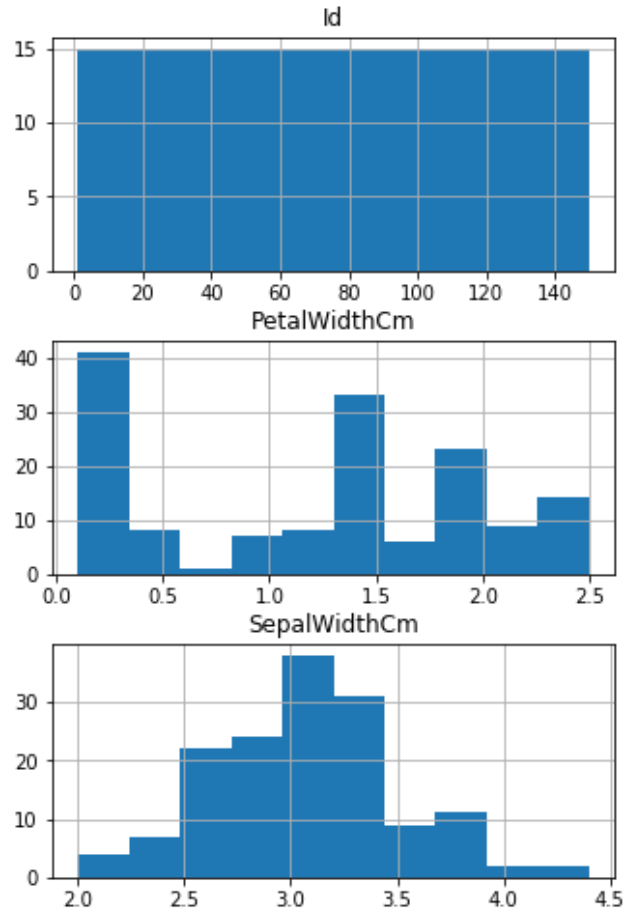
```
                Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  150.000000     150.000000    150.000000     150.000000    150.000000
mean    75.500000       5.843333      3.054000       3.758667      1.198667
std     43.445368       0.828066      0.433594       1.764420      0.763161
min      1.000000       4.300000      2.000000       1.000000      0.100000
25%     38.250000       5.100000      2.800000       1.600000      0.300000
50%     75.500000       5.800000      3.000000       4.350000      1.300000
75%    112.750000       6.400000      3.300000       5.100000      1.800000
max    150.000000       7.900000      4.400000       6.900000      2.500000
```

```
#class distribution - how many records in each class
print(df.groupby('Species').size())
```

```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

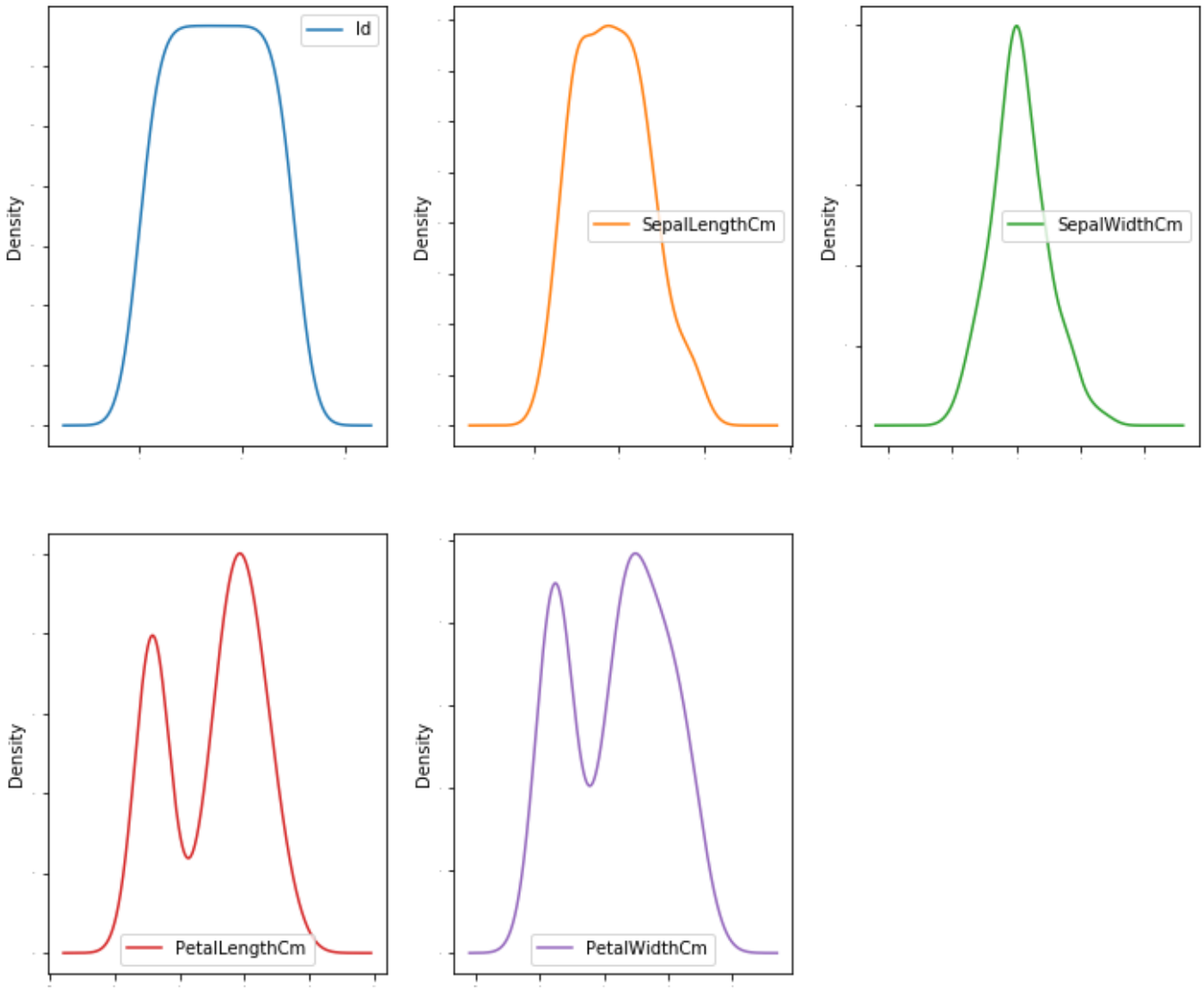
Histogram

```
#plot histogram of each numeric variable / attribute in the data set
df.hist(figsize=(12, 8))
pyplot.show()
```



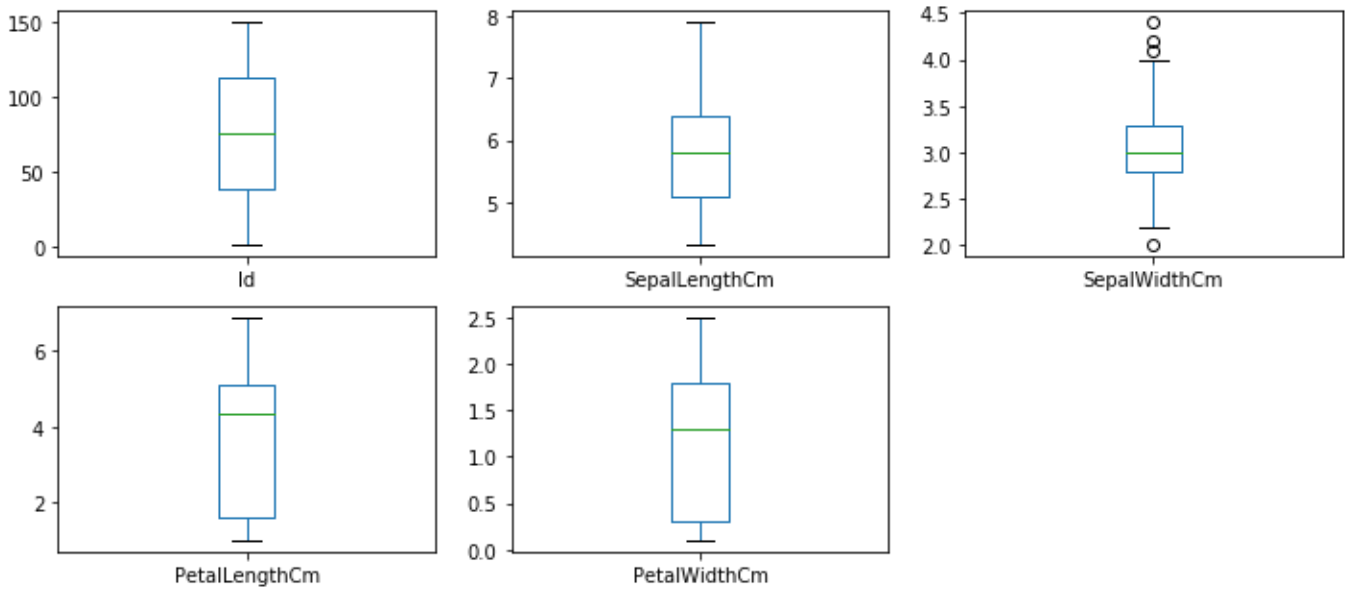
Density Plots

```
# generate density plots of each numeric variable / attribute in the data set
df.plot(kind='density', subplots=True, layout=(3, 3), sharex=False, legend=True,
fontsize=1, figsize=(12, 16))
pyplot.show()
```



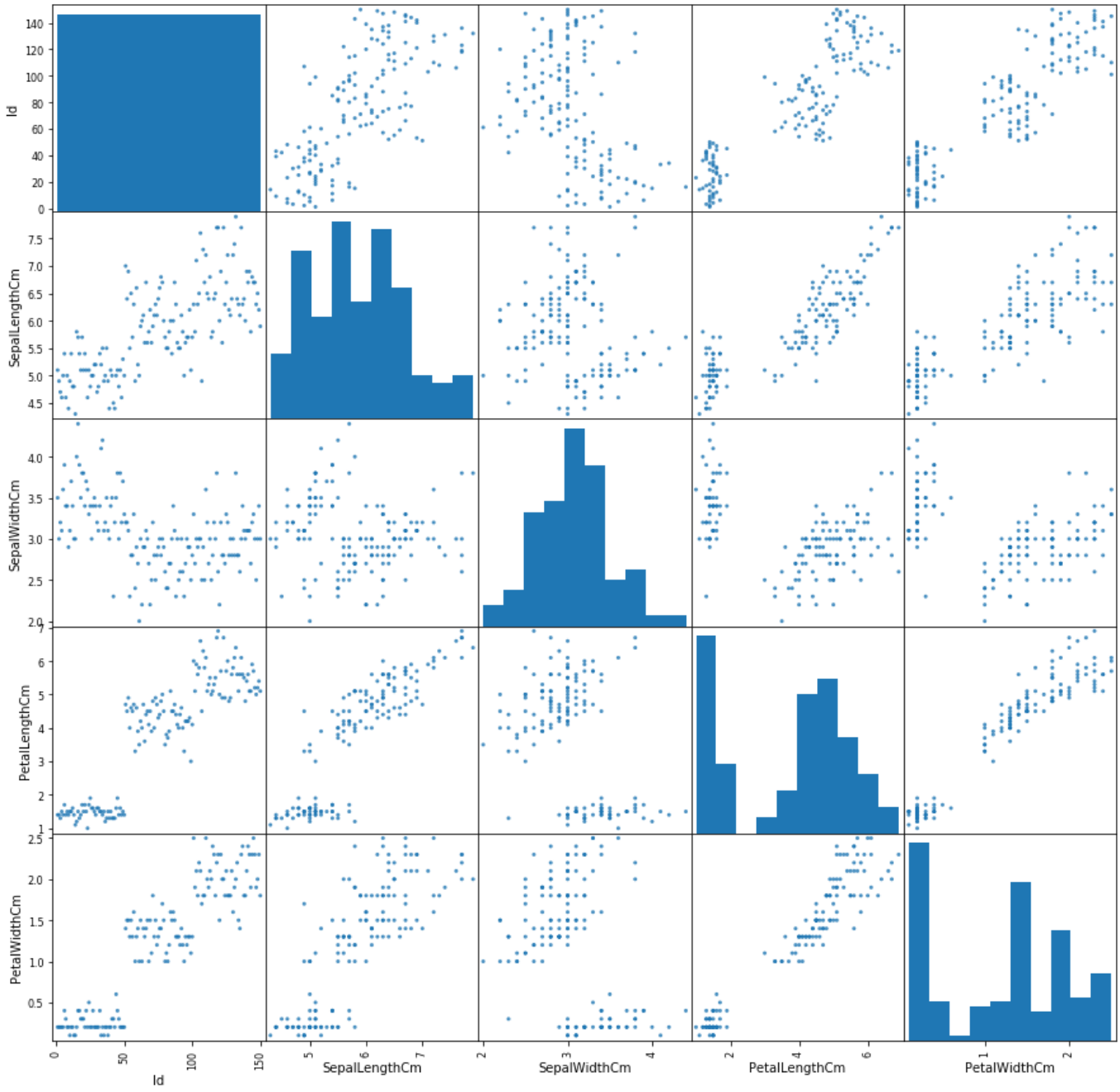
Box Plots

```
# generate box plots of each numeric variable / attribute in the data set
df.plot(kind='box', subplots=True, layout=(3,3), sharex=False, figsize=(12,8))
pyplot.show()
```



Scatter Plots

```
# generate scatter plot matrix of each numeric variable / attribute in the data set
scatter_matrix(df, alpha=0.8, figsize=(15, 15))
pyplot.show()
```



Separate Dataset into Input & Output NumPy arrays

```
# store dataframe values into a numpy array
array = df.values

# separate array into input and output by slicing
# for X(input) [:, 1:5] --> all the rows, columns from 1 - 4 (5 - 1)
# these are the independent variables or predictors
# we will only use this going forward
X = array[:,1:5]

# for Y(input) [:, 5] --> all the rows, column 5
# this is the value we are trying to predict
# we wont use this going forward
Y = array[:,5]
```

Build and Train the Model

```
# Build the model
# set cluster (K) to 3 to start
model = KMeans(n_clusters=3)

# defaults
KMeans(algorithm='auto', copy_x=True, init= 'k-means++', max_iter=300,
n_clusters=3, n_init=10, n_jobs=1, precompute_distances='auto' ,
random_state=None, tol=0.0001, verbose=0)


# Use the model to cluster the inputdata
model.fit (X)

centroids = model.cluster_centers_
print(centroids)
```

```
[[5.9016129  2.7483871  4.39354839 1.43387097]
 [5.006      3.418      1.464      0.244     ]
 [6.85       3.07368421 5.74210526 2.07105263]]
```

```
cluster_labels = model.labels_[::10]
print (cluster_labels)
# this will print 10 records
```

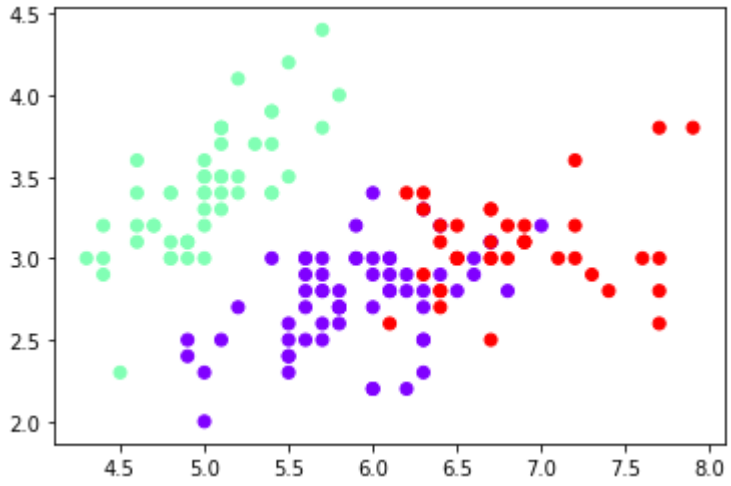
```
[1 1 1 1 1 0 0 0 0 0 2 2 2 2 2]
```

```
cluster_labels = model.labels_
print (cluster_labels)
# this will print all records - shows clustering of values in an unlabeled set
```

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 2 2 2 0 2 2 2 2
 2 2 0 0 2 2 2 2 0 2 0 2 0 2 2 0 0 2 2 2 2 2 0 2 2 2 2 0 2 2 2 0 2 2 2 0 2
 2 0]
```

```
pyplot.scatter(X[:, 0], X[:, 1], c=model.labels_, cmap= 'rainbow' )

pyplot.show ( )
```



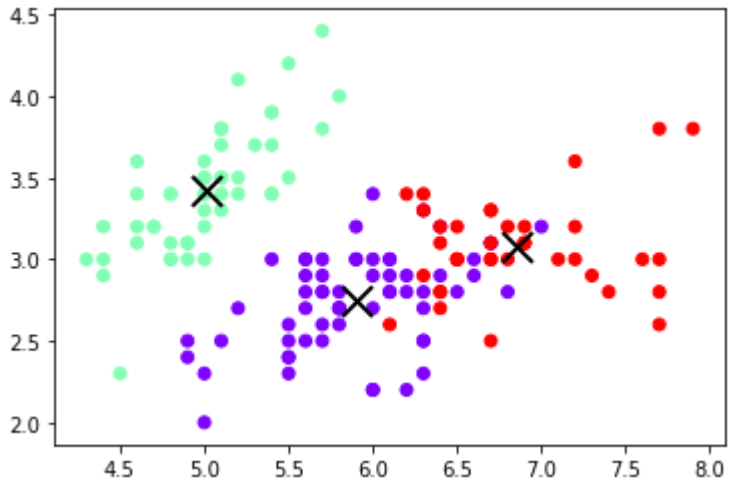

```
# plot the data points with centroids
# plot using first and second variables of the vector
pyplot.scatter(X[:, 0], X[:, 1], c=model.labels_, cmap= 'rainbow')

lines = pyplot.plot(centroids[0,0],centroids[0,1], 'kx', color= 'black')
pyplot.setp (lines, ms=15.0)
pyplot.setp(lines, mew=2.0)

lines = pyplot.plot(centroids[1,0],centroids[1,1], 'kx', color= 'black')
pyplot.setp (lines, ms=15.0)
pyplot.setp(lines, mew=2.0)

lines = pyplot.plot(centroids[2,0],centroids[2,1], 'kx', color= 'black')
pyplot.setp (lines, ms=15.0)
pyplot.setp(lines, mew=2.0)

pyplot.show ()
```



Classify/Predict Model

```
model.predict([[5.3, 3.0, 4.5, 1.5]])
```

```
array([0])
```