

## Task - 2 : Prediction using Unsupervised ML

### K-Means Clustering

In this section we will see how the Python and Scikit-Learn Library for Machine Learning can be used to implement KMeans Clustering functions. Let's start with simple KMeans Clustering involving Four(4) variables named SepalLength, SepalWidth, Petal.Length and Petal.Width.

#### Simple K-Means Clustering

In this Clustering Task we are going to predict the optimum number of clusters. This is a simple K-Means Clustering task.

Author: Vedant Pandya

```
In [1]: #Let's import the libraries we are going to use.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets
print("Libraries are imported.")

Libraries are imported.

In [2]: #Import the data from the CSV File
iris = pd.read_csv("Task_2_data.csv")
print("Dataset imported Successfully!")
iris.head(10)

Dataset imported Successfully!

Out[2]:
```

	Id	SepalLengthCm	SepalWidthCm	Petal.LengthCm	Petal.WidthCm	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
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

```


In [3]: iris.drop('Id', inplace = True, axis = 1)

In [4]: iris.head(10)

Out[4]:
```

	SepalLengthCm	SepalWidthCm	Petal.LengthCm	Petal.WidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

```


In [5]: len(iris)

Out[5]: 150

In [6]: len(iris.columns)

Out[6]: 5

In [7]: iris.shape

Out[7]: (150, 5)

In [8]: iris.columns

Out[8]: Index(['SepalLengthCm', 'SepalWidthCm', 'Petal.LengthCm', 'Petal.WidthCm',
              'Species'],
              dtype='object')

In [9]: print('Dataset Info: {}'.format(iris.info()))

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype  
---  --
 0   SepalLengthCm      150 non-null    float64
 1   SepalWidthCm       150 non-null    float64
 2   Petal.LengthCm     150 non-null    float64
 3   Petal.WidthCm      150 non-null    float64
 4   Species            150 non-null    object  
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
Dataset Info: None

In [10]: print('Dataset Description: \n\n{}'.format(iris.describe()))

Dataset Description:

      SepalLengthCm  SepalWidthCm  Petal.LengthCm  Petal.WidthCm
count  150.000000      150.000000      150.000000      150.000000
mean     5.843333       3.054000       3.758667       1.198667
std       0.520665       0.435304       1.764420       0.763161
min       4.300000       2.000000       1.000000       0.100000
25%       5.100000       2.000000       1.000000       0.300000
50%       5.800000       3.000000       4.350000       1.300000
75%       6.400000       3.300000       5.100000       1.800000
max       7.900000       4.400000       6.900000       2.500000

In [11]: iris.nunique()

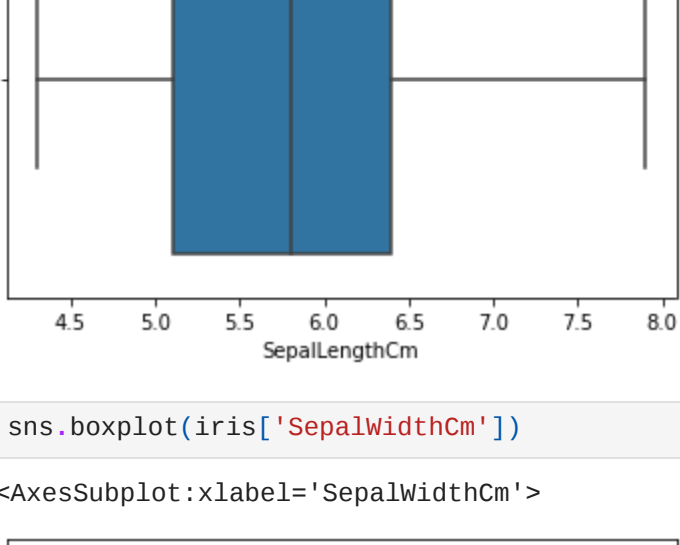
Out[11]: SepalLengthCm    35
SepalWidthCm         23
Petal.LengthCm       42
Petal.WidthCm        22
Species              3
dtype: int64

In [12]: import warnings
warnings.filterwarnings('ignore')

Visualising the Data through the BoxPlot

In [13]: sns.boxplot(iris['Sepal.LengthCm'])

Out[13]: <AxesSubplot:xlabel='Sepal.LengthCm'>
```



```
In [14]: sns.boxplot(iris['Sepal.WidthCm'])

Out[14]: <AxesSubplot:xlabel='Sepal.WidthCm'>
```

A box plot showing the distribution of Sepal.WidthCm. The x-axis is labeled 'Sepal.WidthCm' and ranges from 2.0 to 4.5. The plot shows a median around 3.05, with the IQR spanning from approximately 2.0 to 3.4. Whiskers extend from 2.0 to 4.4, and there are no outliers.

```
In [15]: sns.boxplot(iris['Petal.LengthCm'])

Out[15]: <AxesSubplot:xlabel='Petal.LengthCm'>
```

A box plot showing the distribution of Petal.LengthCm. The x-axis is labeled 'Petal.LengthCm' and ranges from 1 to 7. The plot shows a median around 4.35, with the IQR spanning from approximately 3.7 to 5.1. Whiskers extend from 1.0 to 6.9, and there are no outliers.

```
In [16]: sns.boxplot(iris['Petal.WidthCm'])

Out[16]: <AxesSubplot:xlabel='Petal.WidthCm'>
```

A box plot showing the distribution of Petal.WidthCm. The x-axis is labeled 'Petal.WidthCm' and ranges from 0.0 to 2.5. The plot shows a median around 1.19, with the IQR spanning from approximately 0.3 to 1.8. Whiskers extend from 0.1 to 2.5, and there are no outliers.

BoxPlot are usually used in Exploratory Data Analysis (EDA) for specifically indicating whether there are any potential unusual observation or outliers present in the data set or not.

```
In [17]: def outlier_detect(iris):
        for i in iris.describe().columns:
            Q1 = iris.describe().at['25%', i]
            Q3 = iris.describe().at['75%', i]
            IQR = Q3 - Q1
            LTV = Q1 - 1.5 * IQR
            UTV = Q3 + 1.5 * IQR
            iris[i] = iris[i].mask(iris[i] < LTV, LTV)
            iris[i] = iris[i].mask(iris[i] > UTV, UTV)
        return iris

In [18]: iris = outlier_detect(iris)

Out[18]:
```

	Sepal.LengthCm	Sepal.WidthCm	Petal.LengthCm	Petal.WidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows x 5 columns

```
In [19]: iris.isnull().sum()

Out[19]: Sepal.LengthCm    0
Sepal.WidthCm          0
Petal.LengthCm         0
Petal.WidthCm          0
Species                0
dtype: int64
```

```
In [20]: iris['Species'].value_counts()

Out[20]: Iris-virginica    50
Iris-setosa              50
Iris-versicolour         50
Name: Species, dtype: int64
```

```
In [21]: iris.corr()

Out[21]:
```

	Sepal.LengthCm	Sepal.WidthCm	Petal.LengthCm	Petal.WidthCm
Sepal.LengthCm	1.000000	-0.110343	0.871754	0.817954
Sepal.WidthCm	-0.110343	1.000000	-0.419823	-0.355882
Petal.LengthCm	0.871754	-0.419823	1.000000	0.962757
Petal.WidthCm	0.817954	-0.355882	0.962757	1.000000

```
In [22]: iris1 = iris.corr()
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(iris1, cmap = 'coolwarm', vmin = -1, vmax = 1)
fig.colorbar(cax)
ticks = np.arange(0, len(iris.columns), 1)
ax.set_xticks(ticks)
plt.xticks(rotation = 90)
ax.set_yticks(ticks)
ax.set_xticklabels(iris.columns)
ax.set_yticklabels(iris.columns)

Out[22]: [Text(0, 0, 'Sepal.LengthCm'),
Text(0, 1, 'Sepal.WidthCm'),
Text(0, 2, 'Petal.LengthCm'),
Text(0, 3, 'Petal.WidthCm'),
Text(0, 4, 'Species')]

Sepal.LengthCm
Sepal.WidthCm
Petal.LengthCm
Petal.WidthCm
Species
```

A heatmap showing the correlation matrix of the Iris dataset features. The x and y axes are labeled with the feature names: Sepal.LengthCm, Sepal.WidthCm, Petal.LengthCm, Petal.WidthCm, and Species. The color scale ranges from -1.00 (blue) to 1.00 (red). The diagonal elements are all 1.00 (red). The off-diagonal elements show the correlation between pairs of features: Sepal.LengthCm and Petal.LengthCm have a high positive correlation (0.87), while Sepal.LengthCm and Sepal.WidthCm have a low negative correlation (-0.11).

```
In [23]: fig, ax = plt.subplots(figsize = (10, 10))
sns.heatmap(iris1, vmin = 0, vmax = 1, square = True, annot = True, linewidth = 1)

Out[23]: <AxesSubplot:>
```

An annotated heatmap showing the correlation matrix of the Iris dataset features. The x and y axes are labeled with the feature names: Sepal.LengthCm, Sepal.WidthCm, Petal.LengthCm, and Petal.WidthCm. The color scale ranges from 0.0 (dark purple) to 1.0 (dark red). The diagonal elements are all 1.0. The off-diagonal elements show the correlation between pairs of features: Sepal.LengthCm and Petal.LengthCm have a high positive correlation (0.87), while Sepal.LengthCm and Sepal.WidthCm have a low negative correlation (-0.11).

```
In [24]: sns.set(style = "ticks", color_codes = True)
iris = sns.load_dataset("Iris")
g = sns.pairplot(iris)
plt.show()

Out[24]:
```

A pair plot showing the distribution and relationships between the four features of the Iris dataset: sepal\_length, sepal\_width, petal\_length, and petal\_width. The diagonal shows histograms of each feature, and the off-diagonal shows scatter plots of each pair of features. The plot is styled with 'ticks' and 'color\_codes'.

```
In [25]: pd.DataFrame.drop_duplicates(iris)

Out[25]:
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

149 rows x 5 columns

#### Finding the optimum number of Clusters for K\_Means Classification

```
In [26]: x = iris.iloc[:, [0, 1, 2, 3]] values
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)

In [27]: plt.plot(range(1, 11), wcss, color = "green")
plt.title('The Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()

The Elbow Method
```

A line plot showing the Within-Cluster Sum of Squares (WCSS) on the y-axis (ranging from 0 to 700) versus the Number of Clusters on the x-axis (ranging from 1 to 10). The plot shows a sharp decrease in WCSS from 1 to 4 clusters, after which it levels off. The plot is titled 'The Elbow Method' and has 'Number of Clusters' on the x-axis and 'WCSS' on the y-axis.

```
In [28]: kmeans = KMeans(n_clusters = 3, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
y_kmeans = kmeans.fit_predict(x)

In [29]: plt.figure(figsize=(15,10))
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Iris-setosa')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Iris-versicolour')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Iris-virginica')
plt.scatter(kmeans.cluster_centers[:, 0], kmeans.cluster_centers[:, 1], s = 100, c = 'yellow', label = 'Centroids')
plt.legend()

Out[29]: <matplotlib.legend.Legend at 6x1b77c17bee>
```

A scatter plot showing the distribution of the Iris dataset with 3 clusters and their centroids. The x-axis ranges from 4.5 to 8.0 and the y-axis ranges from 2.0 to 4.5. The clusters are colored red (Iris-setosa), blue (Iris-versicolour), and green (Iris-virginica). The centroids are marked with yellow dots. A legend is shown in the top right corner.

Thank You!