**The Sparks Foundation - GRIP - Data Science and Business Analytics Intern - AUGUST-2021**

# TASK 2 - Prediction the optimum number of clusters From given iris dataset

by JANGAM RANJEETH

DATASET LINK-[https://bit.ly/3cGyP8j (https://bit.ly/3cGyP8j)](https://bit.ly/3cGyP8j)

In this task we are going predict optimum number of clusters formation and visualize it using Elbow method

# Step1 Defining objectives

In

[12]:

*#importing nessessary libraries*

**import**

sklearn

**import**

numpy

**as**

np

**import**

pandas

**as**

pd

**import**

matplotlib

.

pyplot

**as**

plt

**import**

seaborn

**as**

sn

**import**

warnings

warnings

.

filterwarnings

(

'ignore'

)

# Step2 Data collection

[13]:

Out[13]:

**Id**

**SepalLengthCm**

**SepalWidthCm**

**PetalLengthCm**

**PetalWidthCm**

**Species**

**0**

1

5.1

3.5

1.4

0.2

Iris-setosa

*#importing the dataset and displaying*

dt

**=**

pd

.

read\_csv

(

"Iris.csv"

)

dt

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. 146 6.7 3.0 5.2 2.3 Iris-virginica
6. 147 6.3 2.5 5.0 1.9 Iris-virginica
7. 148 6.5 3.0 5.2 2.0 Iris-virginica
8. 149 6.2 3.4 5.4 2.3 Iris-virginica
9. 150 5.9 3.0 5.1 1.8 Iris-virginica
10. rows × 6 columns

# Step3 Data Preprocessing

In

[14]:

Out[14]:

**Id**

**SepalLengthCm**

**SepalWidthCm**

**PetalLengthCm**

**PetalWidthCm**

**count**

150.000000

150.000000

150.000000

150.000000

150.000000

dt

.

describe

()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 |

[15]:

dt

.

info

()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 150 entries, 0 to 149

Data columns (total 6 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

1. Id 150 non-null int64
2. SepalLengthCm 150 non-null float64
3. SepalWidthCm 150 non-null float64
4. PetalLengthCm 150 non-null float64
5. PetalWidthCm 150 non-null float64
6. Species 150 non-null object dtypes: float64(4), int64(1), object(1) memory usage: 7.2+ KB

In [16]: print(dt.isnull().sum(),'\n\n Number of duplicate rows:',dt.duplicated().sum())

Id 0

SepalLengthCm 0

SepalWidthCm 0

PetalLengthCm 0

PetalWidthCm 0 Species 0 dtype: int64

In

[17]:

Number of duplicate rows: 0

*#Removing the duplicates*

dt

.

drop\_duplicates

(

inplace

**=**

**True**

)

dt

.

shape

[

0

]

Out[17]: 150

In

[18]:

*#removing the id column*

dt

**=**

dt

.

iloc

[:

,

1

:]

dt

.

columns

Out[18]: Index(['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',

'Species'], dtype='object')

# Step4 Data divided into clusters

[19]:

x

**=**

dt

.

iloc

[:

,

[

0

,

1

,

2

]].

values

**from**

sklearn

.

cluster

**import**

KMeans

km

**=**

KMeans

(

n\_clusters

**=**

3

)

km

.

fit

(

x

)

Out[19]: KMeans(n\_clusters=3)

In

[20]:

km

.

cluster\_centers\_

*#finding nearest values*

Out[20]: array([[5.006 , 3.418 , 1.464 ], [6.83571429, 3.06428571, 5.6547619 ],

[5.84655172, 2.73275862, 4.3637931 ]])

In

[21]:

*#data is labeled as centroid values*

pred

**=**

km

.

labels\_

pred

Out[21]: array([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, 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, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,

2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2,

2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1,

1, 1, 1, 2, 2, 1, 1, 1, 1, 2, 1, 2, 1, 2, 1, 1, 2, 2, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 2])

In

[22]:

Out[22]:

**SepalLengthCm**

**SepalWidthCm**

**PetalLengthCm**

**PetalWidthCm**

**Species**

**clusters**

**0**

5.1

3.5

1.4

0.2

Iris-setosa

0

dt

[

'clusters'

]

**=**

pred

dt

1. 4.9 3.0 1.4 0.2 Iris-setosa 0
2. 4.7 3.2 1.3 0.2 Iris-setosa 0
3. 4.6 3.1 1.5 0.2 Iris-setosa 0
4. 5.0 3.6 1.4 0.2 Iris-setosa 0

**...** ... ... ... ... ... ...

1. 6.7 3.0 5.2 2.3 Iris-virginica 1
2. 6.3 2.5 5.0 1.9 Iris-virginica 2
3. 6.5 3.0 5.2 2.0 Iris-virginica 1
4. 6.2 3.4 5.4 2.3 Iris-virginica 1
5. 5.9 3.0 5.1 1.8 Iris-virginica 2
6. rows × 6 columns

[23]: display(dt['clusters'].value\_counts(),dt['Species'].value\_counts())

2 58

1. 50
2. 42

Name: clusters, dtype: int64

Iris-versicolor 50

Iris-setosa 50

Iris-virginica 50

Name: Species, dtype: int64

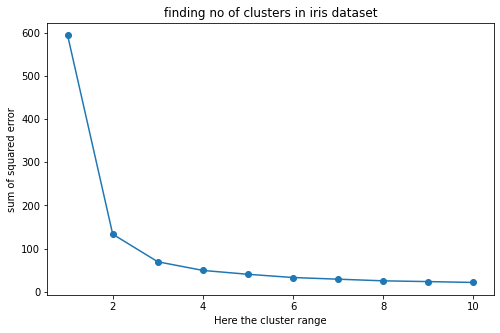
# Step 5 Prediction using Elbow method

In

[24]:

In

[28]:



*#finding optimum number of clusters*

wss

**=**

[]

cluster\_range

**=**

range

(

1

,

11

)

**for**

k

**in**

cluster\_range

:

km

**=**

KMeans

(

n\_clusters

**=**

k

,

random\_state

**=**

0

)

km

.

fit

(

x

)

inertia

**=**

km

.

inertia\_

wss

.

append

(

inertia

)

plt

.

figure

(

figsize

**=**

(

8

,

5

))

plt

.

xlabel

(

"Here the cluster range"

)

plt

.

ylabel

(

"sum of squared error"

)

plt

.

title

(

"finding no of clusters in iris dataset"

)

plt

.

plot

(

cluster\_range

,

wss

,

marker

**=**

"o"

)

plt

.

show

()

# Step 6 Visualization of clusters

In [26]: *#fitting the data* kmeans **=** KMeans(n\_clusters **=** 3, init **=** 'k-means++', max\_iter **=** 300, n\_init **=** 10, random\_state **=** 0) y\_kmeans **=** kmeans.fit\_predict(x)

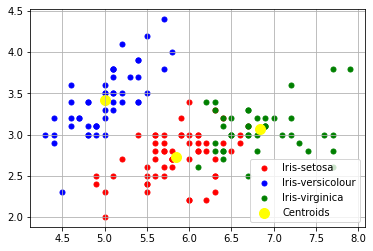
In [27]: *# Visualising the clusters - On the first two columns* plt.scatter(x[y\_kmeans **==** 0, 0], x[y\_kmeans **==** 0, 1], s **=** 25, c **=** 'red', label **=** plt.scatter(x[y\_kmeans **==** 1, 0], x[y\_kmeans **==** 1, 1], s **=** 25, c **=** 'blue', label **=** plt.scatter(x[y\_kmeans **==** 2, 0], x[y\_kmeans **==** 2, 1],s **=** 25, c **=** 'green', label **=**

*# Plotting the centroids of the clusters*

plt.scatter(kmeans.cluster\_centers\_[:, 0],kmeans.cluster\_centers\_[:,1], s **=** 100, c **=** 'yellow', label **=** 'Centroids') plt.grid() plt.legend()

Out[27]: <matplotlib.legend.Legend at 0x15f1b6487c0>

In [ ]:



In [ ]: