Project Report

Image Segmentation

Name:Keshav Anand

Course: AI and ML

(Batch-4)

Duration: 12 months

Problem Statement: Image segmentation using K-Means.

Prerequisites

What things you need to install the software and how to install them:

Python 3.6 This setup requires that your machine has latest version of python. The following url https://www.python.org/downloads/ can be referred to download python. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this: https://www.pythoncentral.io/add-python-to-path-python-is-not- recognized-as-an-internal-or-external-command/. Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic.

Second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this url https://www.anaconda.com/download/ You will also need to download and install below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.6 then run below commands in command prompt/terminal to install these packages pip install -U scikit-learn pip install numpy pip install scipy if you have chosen to install anaconda then run below commands in anaconda prompt to install these packages conda install -c scikit-learn conda install -c anaconda numpy conda install -c anaconda scipy

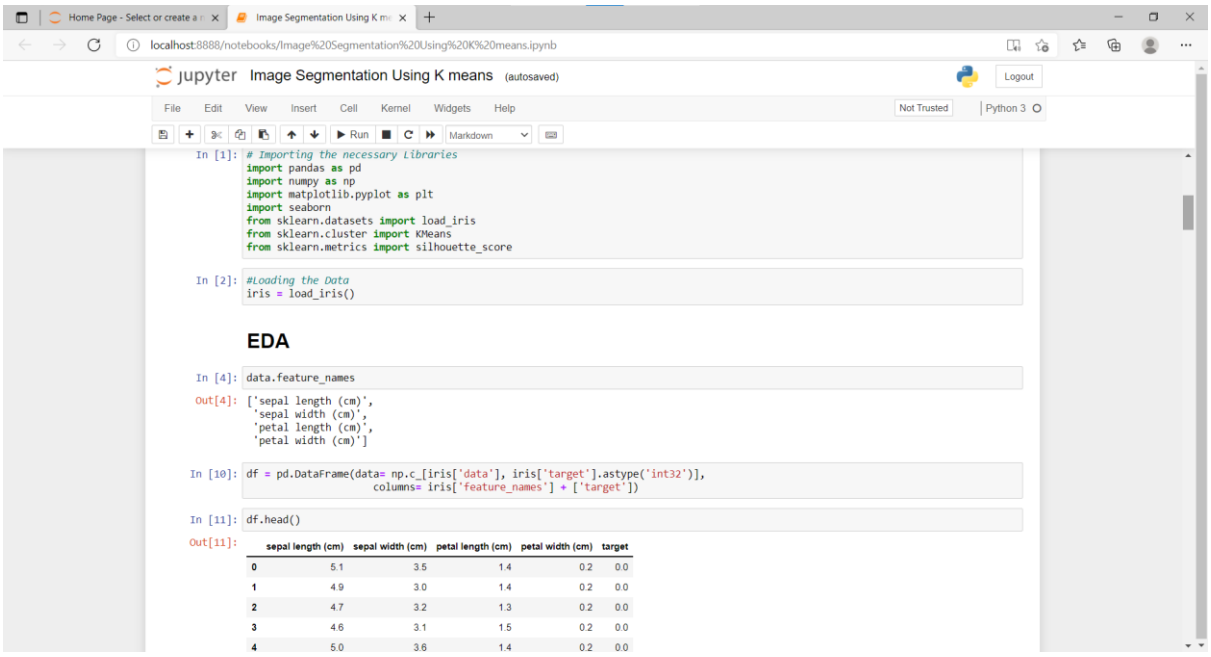
Dataset used:

The dataset used is IRIS dataset which is an in-built dataset available in scikit-learn library and a random photo downloaded from the internet.

Method used for Segmentation:

K-MEANS

Screenshots of Source Code and Output:



Home Page - Select or create a n | Image Segmentation Using K me

localhost:8888/notebooks/Image%20Segmentation%20Using%20K%20means.ipynb

jupyter Image Segmentation Using K means (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run | Markdown

In [1]:
```
# Importing the necessary Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

In [2]:
```
#Loading the Data
iris = load_iris()
```

EDA

In [4]:
```
data.feature_names
```

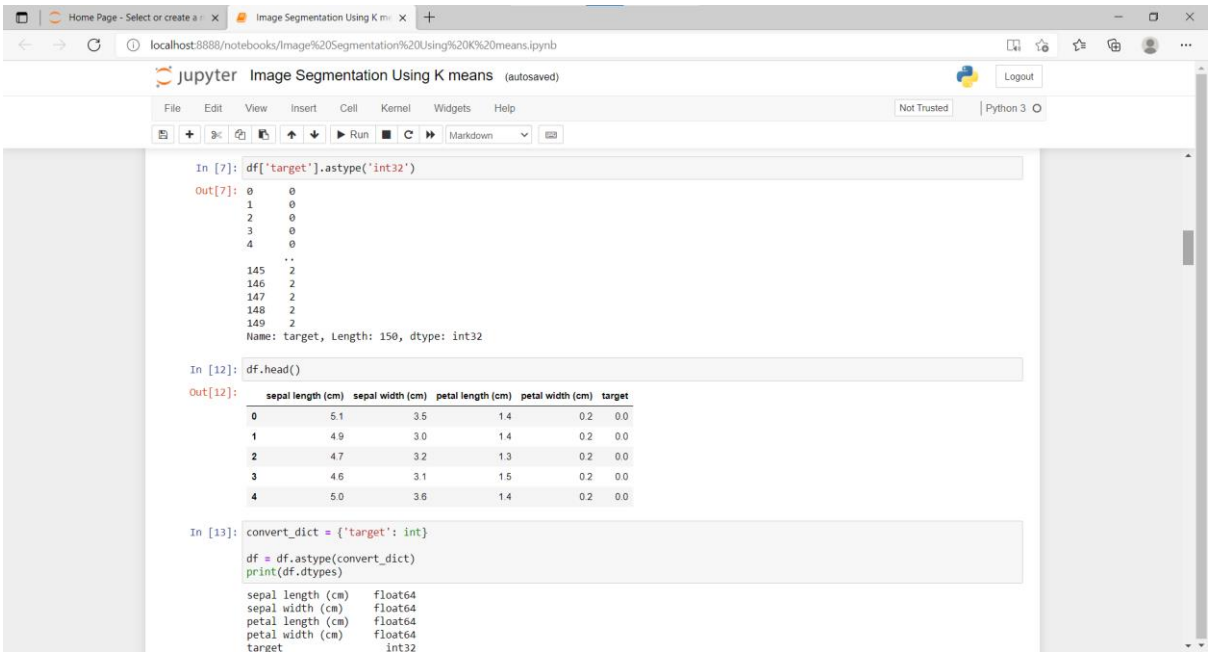
Out[4]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [10]:
```
df = pd.DataFrame(data= np.c_[iris['data'], iris['target'].astype('int32')],
                  columns= iris['feature_names'] + ['target'])
```

In [11]:
```
df.head()
```

Out[11]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0



Home Page - Select or create a n | Image Segmentation Using K me

localhost:8888/notebooks/Image%20Segmentation%20Using%20K%20means.ipynb

jupyter Image Segmentation Using K means (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run | Markdown

In [7]:
```
df['target'].astype('int32')
```

Out[7]:
```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: target, Length: 150, dtype: int32
```

In [12]:
```
df.head()
```

Out[12]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0

In [13]:
```
convert_dict = {'target': int}

df = df.astype(convert_dict)
print(df.dtypes)
```

```
sepal length (cm)    float64
sepal width (cm)     float64
petal length (cm)    float64
petal width (cm)     float64
target                 int32
```

Home Page - Select or create a n | Image Segmentation Using K me

localhost:8888/notebooks/Image%20Segmentation%20Using%20K%20means.ipynb

jupyter Image Segmentation Using K means (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Markdown

In [15]:
```
df.head()
```

Out[15]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [19]:
```
print(df.describe())
```

```
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)      target
count        150.000000  150.000000
mean           1.199333    1.000000
std            0.762238    0.819232
min            0.100000    0.000000
25%            0.300000    0.000000
50%            1.300000    1.000000
75%            1.800000    2.000000
max            2.500000    2.000000
```

In [20]:
```
iris.target_names
```

Out[20]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

Home Page - Select or create a n | Image Segmentation Using K me

localhost:8888/notebooks/Image%20Segmentation%20Using%20K%20means.ipynb

jupyter Image Segmentation Using K means (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Markdown

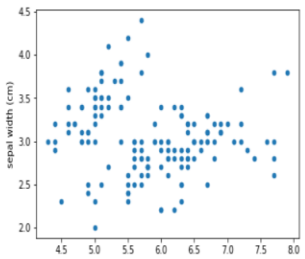
In [24]:
```
df.info()
```

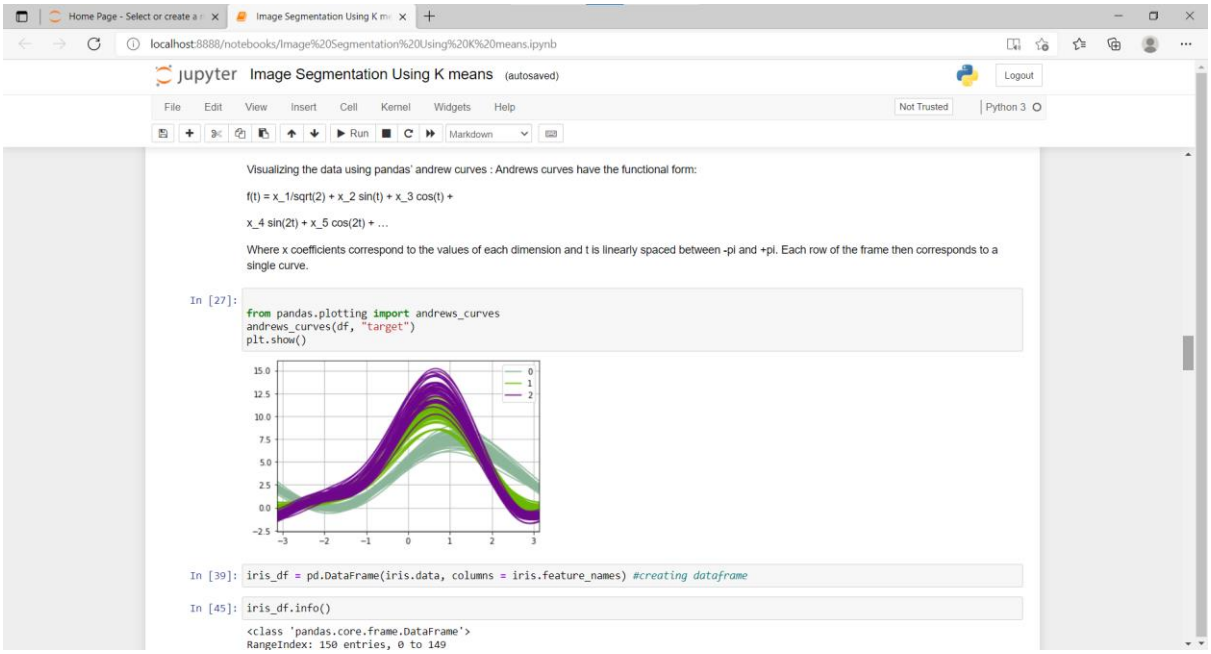
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   target             150 non-null    int32  
dtypes: float64(4), int32(1)
memory usage: 5.4 KB
```

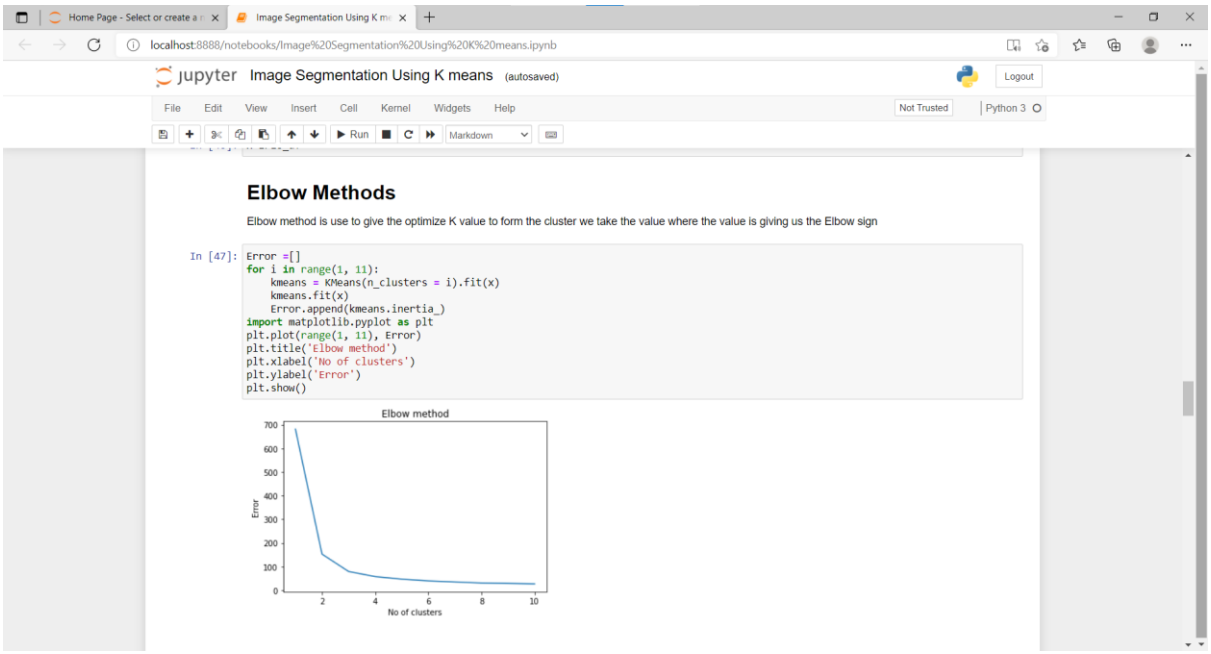
Visulizing the Data

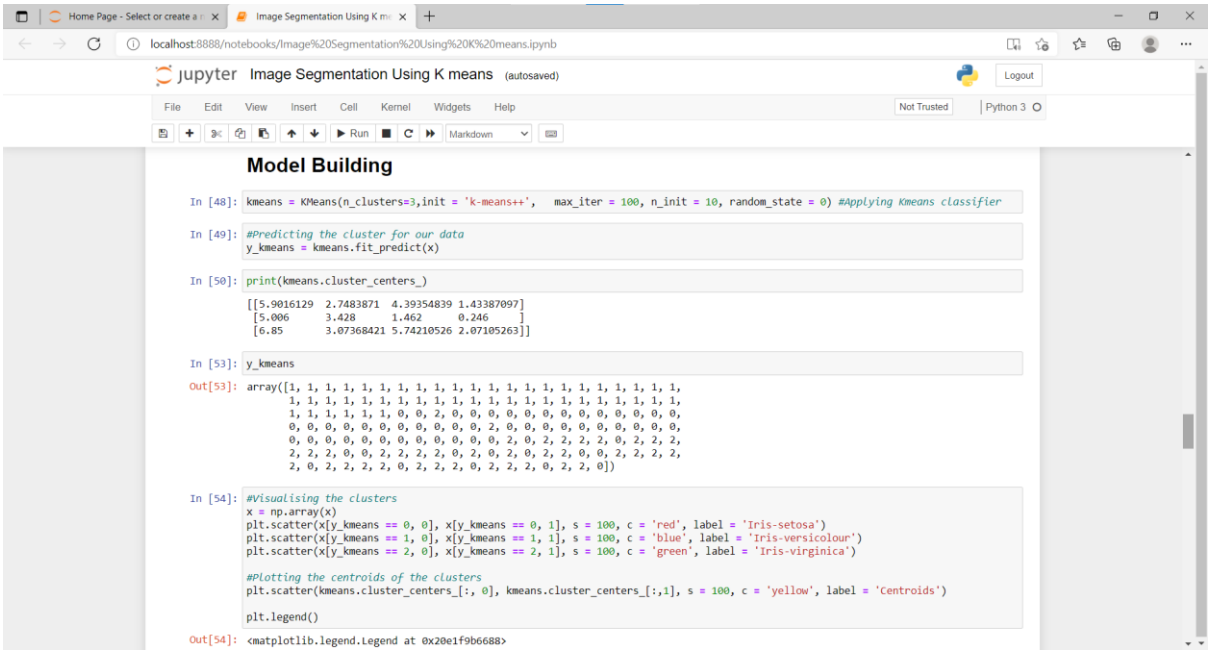
In [25]:
```
df.plot(kind='scatter',x = 'sepal length (cm)',y = 'sepal width (cm)')
```

Out[25]:
```
<AxesSubplot:xlabel='sepal length (cm)', ylabel='sepal width (cm)'>
```




Home Page - Select or create a
Image Segmentation Using K m
localhost:8888/notebooks/Image%20Segmentation%20Using%20K%20means.ipynb
jupyter Image Segmentation Using K means (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
Visualizing the data using pandas' andrew curves : Andrews curves have the functional form:
f(t) = x_1/sqrt(2) + x_2 sin(t) + x_3 cos(t) +
x_4 sin(2t) + x_5 cos(2t) + …
Where x coefficients correspond to the values of each dimension and t is linearly spaced between -pi and +pi. Each row of the frame then corresponds to a single curve.
In [27]:
from pandas.plotting import andrews_curves
andrews_curves(df, "target")
plt.show()
0
1
2
In [39]:
iris_df = pd.DataFrame(iris.data, columns = iris.feature_names) #creating dataframe
In [45]:
iris_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149


Home Page - Select or create a
Image Segmentation Using K m
localhost:8888/notebooks/Image%20Segmentation%20Using%20K%20means.ipynb
jupyter Image Segmentation Using K means (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
Elbow Methods
Elbow method is use to give the optimize K value to form the cluster we take the value where the value is giving us the Elbow sign
In [47]:
Error =[]
for i in range(1, 11):
kmeans = KMeans(n_clusters = i).fit(x)
kmeans.fit(x)
Error.append(kmeans.inertia_)
import matplotlib.pyplot as plt
plt.plot(range(1, 11), Error)
plt.title('Elbow method')
plt.xlabel('No of clusters')
plt.ylabel('Error')
plt.show()
Elbow method
Error
No of clusters



Model Building

In [48]:
```
kmeans = KMeans(n_clusters=3,init = 'k-means++',   max_iter = 100, n_init = 10, random_state = 0) #Applying Kmeans classifier
```

In [49]:
```
#Predicting the cluster for our data
y_kmeans = kmeans.fit_predict(x)
```

In [50]:
```
print(kmeans.cluster_centers_)
```

```
[[5.9016129  2.7483871  4.39354839 1.43387097]
 [5.006      3.428      1.462      0.246     ]
 [6.85       3.07368421 5.74210526 2.07105263]]
```

In [53]:
```
y_kmeans
```

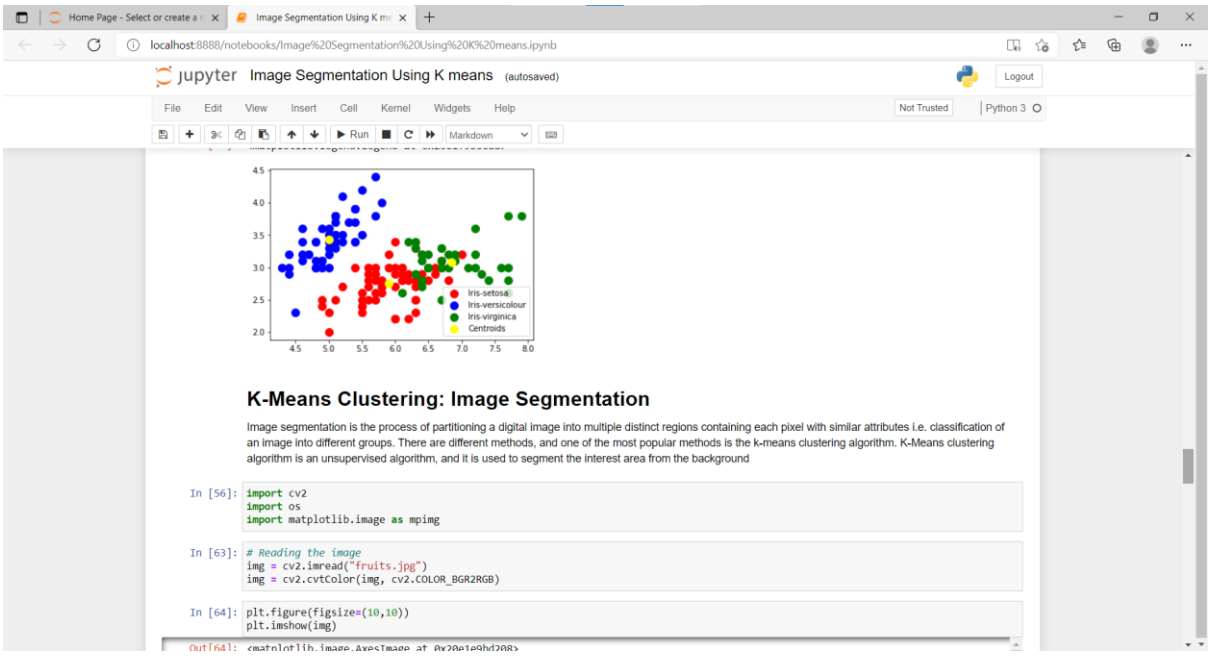
Out[53]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 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])
```

In [54]:
```
#Visualising the clusters
x = np.array(x)
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')

#Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1], s = 100, c = 'yellow', label = 'Centroids')

plt.legend()
```

Out[54]:
```
<matplotlib.legend.Legend at 0x20e1f9b6688>
```



K-Means Clustering: Image Segmentation

Image segmentation is the process of partitioning a digital image into multiple distinct regions containing each pixel with similar attributes i.e. classification of an image into different groups. There are different methods, and one of the most popular methods is the k-means clustering algorithm. K-Means clustering algorithm is an unsupervised algorithm, and it is used to segment the interest area from the background

In [56]:
```
import cv2
import os
import matplotlib.image as mpimg
```

In [63]:
```
# Reading the image
img = cv2.imread("fruits.jpg")
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

In [64]:
```
plt.figure(figsize=(10,10))
plt.imshow(img)
```

Out[64]:
```
<matplotlib.image.AxesImage at 0x20e1e9bd208>
```

Home Page - Select or create a n | Image Segmentation Using K me

localhost:8888/notebooks/Image%20Segmentation%20Using%20K%20means.ipynb

jupyter Image Segmentation Using K means (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run | Markdown

In [56]:
```
import cv2
import os
import matplotlib.image as mpimg
```

In [63]:
```
# Reading the image
img = cv2.imread("fruits.jpg")
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

In [64]:
```
plt.figure(figsize=(10,10))
plt.imshow(img)
```

Out[64]: <matplotlib.figure.AxesImage at 0x20e1e9bd208>

In [62]:
```
img.shape
```

Out[62]: (553, 830, 3)

Home Page - Select or create a n | Image Segmentation Using K me

localhost:8888/notebooks/Image%20Segmentation%20Using%20K%20means.ipynb

jupyter Image Segmentation Using K means (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

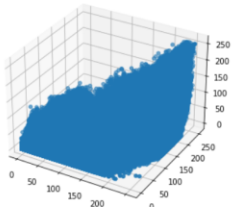
Run | Markdown

Out[62]: (553, 830, 3)

Let us plot an image in 3D space using python matplotlib library.

Below is the image that we'll gonna plot in 3D space

In [58]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import cv2
img = cv2.imread("fruits.jpg")
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
r, g, b = cv2.split(img)
r = r.flatten()
g = g.flatten()
b = b.flatten()#plotting
fig = plt.figure()
ax = Axes3D(fig)
ax.scatter(r, g, b)
plt.show()
```

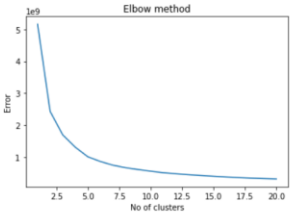


Next, converts the MxNx3 image into a Kx3 matrix where K=MxN and each row is now a vector in the 3-D space of RGB.

In [65]:
```
vectorized = img.reshape((-1,3))
vectorized = np.float32(vectorized)
```

Elbow Method To Determine no of cluster

In [68]:
```
Error =[]
for i in range(1, 21):
    kmeans = KMeans(n_clusters = i).fit(vectorized)
    kmeans.fit(vectorized)
    Error.append(kmeans.inertia_)
import matplotlib.pyplot as plt
plt.plot(range(1, 21), Error)
plt.title('Elbow method')
plt.xlabel('No of clusters')
plt.ylabel('Error')
plt.show()
```



In [70]:
```
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
```

In [71]:
```
K = 7
attempts=10
ret,label,center=cv2.kmeans(vectorized,K,None,criteria,attempts,cv2.KMEANS_PP_CENTERS)
```

In [72]:
```
center = np.uint8(center)
```

In [73]:
```
res = center[label.flatten()]
result_image = res.reshape((img.shape))
```

In [74]:
```
figure_size = 15
plt.figure(figsize=(figure_size,figure_size))
plt.subplot(1,2,1),plt.imshow(img)
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(1,2,2),plt.imshow(result_image)
plt.title('Segmented Image when K = %i' % K), plt.xticks([]), plt.yticks([])
plt.show()
```