Machine Learning Lab 7 - K Means Clustering

Submitted By
Name: **Harsha KG**
Register Number: **19112005**
Class: **5 BSc Data Science**

Lab Overview

Objectives

With 'Iris Dataset':

- Perform K Means Clustering
- Exploratory Data Analysis(EDA)
- Apply K-Means Clustering Algorithm on it.
- Check the performance in terms of both Computation and Accuracy

Problem Definition

- Using K Means Clustering divide the data points into groups that share same similarity metrics
- Perform Exploratory Data Analysis(EDA) with chosen dataset
- Check the performance with respect to change in dataset split, random-state, hyperparameters

Approach

- Preprocess the dataset
- Visualize the dataset before fitting the model
- Split the dataset
- Using K Means Clustering(sklearn.cluster.KMeans) divide the data points into groups
- Fit the model and then predict
- Demonstrate various Evaluation Metrics
- Check the effect of clustering with respect to change in dataset split, random-state, hyperparameters

Sections

1. Lab Overview
2. About the Dataset
3. Importing Libraries
4. Loading the Dataset
5. Data Preprocessing & EDA
6. Dataset Splitting
7. Optimum no.of Clusters
8. Elbow Method
9. K Means
10. Compare the Effect of Different Parameters
11. Observation
12. Conclusion

References

1. https://www.kaggle.com/khotijahs1/k-means-clustering-of-iris-dataset#K-Means (https://www.kaggle.com/khotijahs1/k-means-clustering-of-iris-dataset#K-Means)
2. https://www.analyticsvidhya.com/blog/2021/06/analyzing-decision-tree-and-k-means-clustering-using-iris-dataset/ (https://www.analyticsvidhya.com/blog/2021/06/analyzing-decision-tree-and-k-means-clustering-using-iris-dataset/)
3. https://www.datacamp.com/community/tutorials/k-means-clustering-python (https://www.datacamp.com/community/tutorials/k-means-clustering-python)

4. https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1 (https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1)
5. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html (https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html)
6. https://www.kaggle.com/xvivancos/tutorial-clustering-wines-with-k-means (https://www.kaggle.com/xvivancos/tutorial-clustering-wines-with-k-means)
7. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.bar.html (https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.bar.html)
8. https://github.com/HxnDev/K-Means-on-IRIS-Dataset/blob/main/main.ipynb (https://github.com/HxnDev/K-Means-on-IRIS-Dataset/blob/main/main.ipynb)

About the Dataset

`Iris.csv` consists of 6 attributes:

1. `Id` : ID
2. `SepalLengthCm` : Length of Sepal (cm)
3. `SepalWidthCm` : Width of Sepal (cm)
4. `PetalLengthCm` : Length of Petal (cm)
5. `PetalWidthCm` : Width of Petal (cm)
6. `Species` : Species (Iris-virginica,Iris-setosa,Iris-versicolor)

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
```

Loading the Dataset

In [2]:
```
iris=pd.read_csv('Iris.csv')
```

In [3]:
```
iris.head()
```

Out[3]:

	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

In [4]:
```
iris.tail()
```

Out[4]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

In [5]:
```
iris.sample(5)
```

Out[5]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
52	53	6.9	3.1	4.9	1.5	Iris-versicolor
122	123	7.7	2.8	6.7	2.0	Iris-virginica
95	96	5.7	3.0	4.2	1.2	Iris-versicolor
126	127	6.2	2.8	4.8	1.8	Iris-virginica
29	30	4.7	3.2	1.6	0.2	Iris-setosa

In [6]:
```
iris.shape
```

Out[6]:
```
(150, 6)
```

In [7]:
```
iris.columns
```

Out[7]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

Data Preprocessing & EDA

In [8]:
```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [9]:
```
iris=iris.drop('Id',axis=1)
```

In [10]:
```
iris.shape
```

Out[10]:
```
(150, 5)
```

In [11]:
```
iris.columns
```

Out[11]:
```
Index(['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [12]:
```
iris.describe()
```

Out[12]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	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 [13]:
```
iris.isnull().sum()
```

Out[13]:
```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

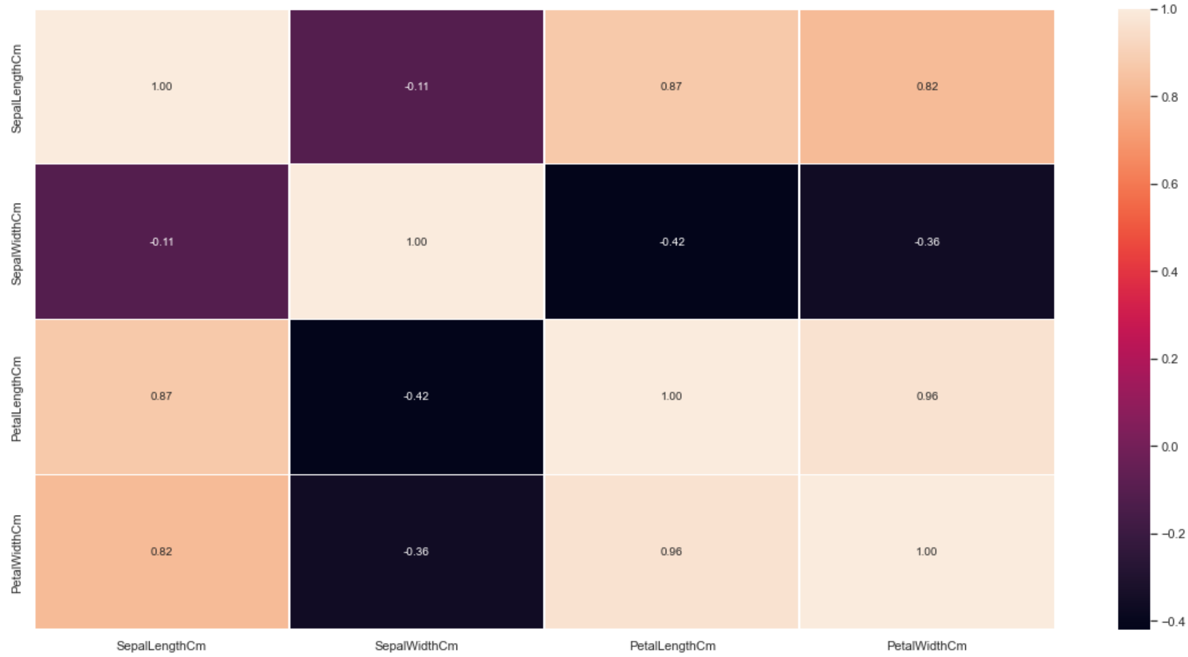
In [69]:
```
iris.corr()
```

Out[69]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

In [68]:
```
plt.figure(figsize=(20,10))
sns.heatmap(iris.corr(),annot=True, fmt=".2f",annot_kws={"size":10},linewidths=.7)
```

Out[68]: <matplotlib.axes._subplots.AxesSubplot at 0x23bdc79b5e0>

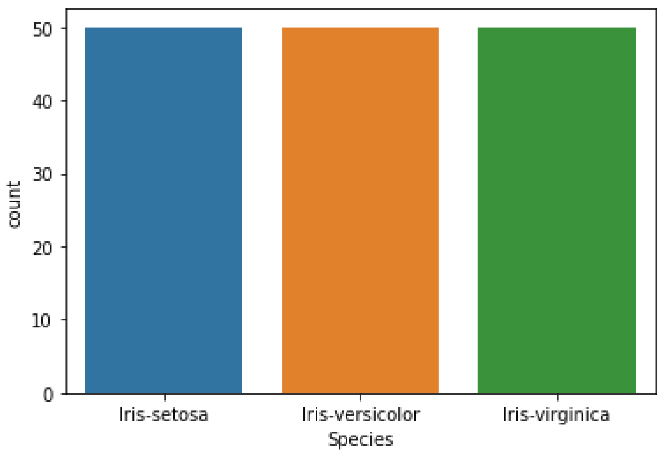


In [14]:
```
iris['Species'].value_counts()
```

Out[14]:
```
Iris-setosa        50
Iris-virginica     50
Iris-versicolor    50
Name: Species, dtype: int64
```

In [15]:
```
sns.countplot(x='Species',data=iris)
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x23bda2746d0>

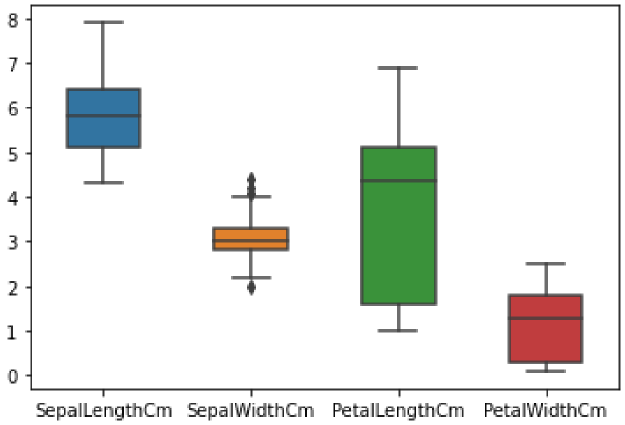


Observation

- Dataset contains 50 samples of 3 species

In [16]:
```
sns.boxplot(data=iris[['SepalLengthCm', 'SepalWidthCm','PetalLengthCm','PetalWidthCm']],
width=0.5,fliersize=5)
```

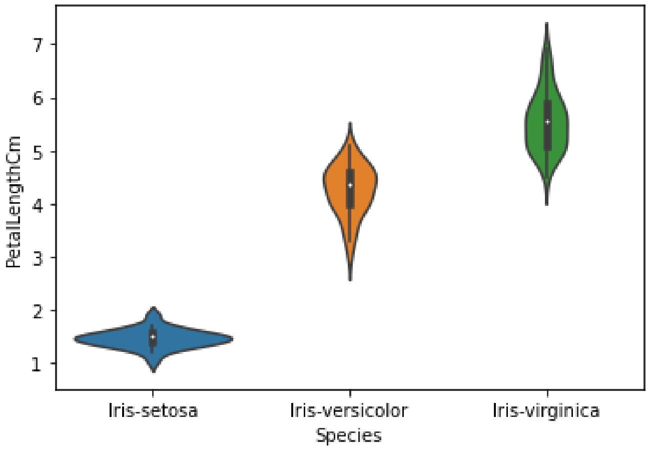
Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x23bda3179a0>
```



Observation

- Plot shows quartiles of length and width of sepal and petal(cm)
- Sepal Width(cm) contains outliers

In [17]:
```
ax = sns.violinplot(x='Species', y='PetalLengthCm', data=iris,size=8)
```



Observation

- Plot shows relationship between Species to Petal Length(cm)
- Median of 'Iris-setosa' species is less than other species

Datset Splitting

In [18]:
```
Y = iris['Species']
X = iris.iloc[:, [0, 1, 2, 3]].values
```

In [19]:
```
X[0:5]
```

Out[19]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2]])
```

In [20]:
```
Y[0:5]
```

Out[20]:
```
0    Iris-setosa
1    Iris-setosa
2    Iris-setosa
3    Iris-setosa
4    Iris-setosa
Name: Species, dtype: object
```

Optimum no. of Clusters

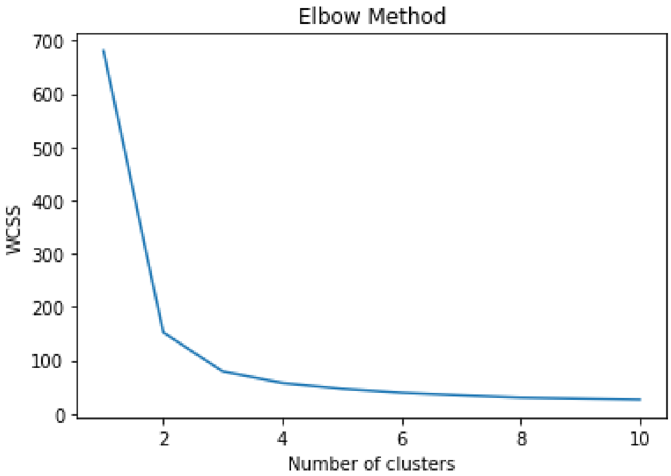
In [21]:
```
# Finding the optimum number of clusters for k-means clustering
from sklearn.cluster import KMeans
wcss = []

for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10, ran
dom_state = 0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
```

In [22]:
```
wcss
```

Out[22]:
```
[680.8243999999996,
 152.36870647733915,
 78.94084142614601,
 57.34540931571815,
 46.535582051282034,
 38.93873974358975,
 34.190687924796634,
 29.90537429982511,
 27.927882157034986,
 25.955497086247092]
```

Elbow Method

In [23]:
```
# Using the elbow method to determine the optimal number of clusters for k-means cluster
ing
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') #within cluster sum of squares
plt.show()
```



Observation

- From the Elbow Method graph, we found that the optimum no.of clusters which can be formed is 3

K Means

In [24]:
```
kmeans = KMeans(n_clusters = 3, init = 'k-means++', max_iter = 300, n_init = 10, random_
state = 42)
y_kmeans = kmeans.fit_predict(X)
```

Fit the model to all the data except for the Species label

In [25]:
```
y_kmeans=kmeans.fit_predict(X)
```

In [26]:
```
print(kmeans.labels_)
```

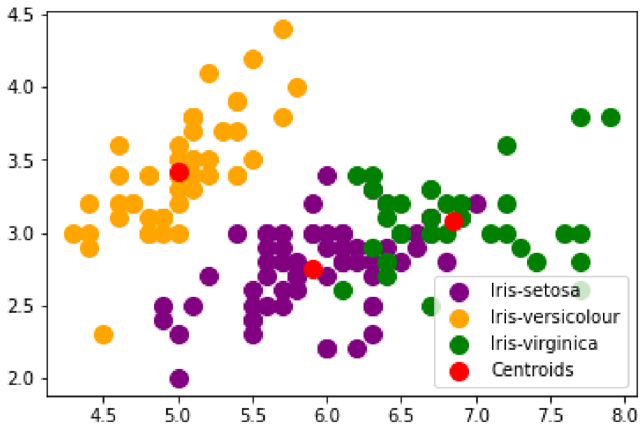
```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 2 2 2 0 2 2 2 2
 2 2 0 0 2 2 2 2 0 2 0 2 0 2 2 0 0 2 2 2 2 2 0 2 2 2 2 0 2 2 2 0 2 2 2 0 2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

In [27]:
```
print(kmeans.cluster_centers_)
```

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

In [28]:
```
#Visualising the clusters
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'purple', label = 'Iris-setosa')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'orange', 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 = 'red', label = 'Centroids')

plt.legend()
```

Out[28]: <matplotlib.legend.Legend at 0x23bdc523490>



In [29]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [30]:
```
#le.fit(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'].sort())

classes = ['Iris-versicolor', 'Iris-setosa', 'Iris-virginica'].sort()

classes
```

In [32]:
```
y_trans=le.fit_transform(Y)
```

In [33]:
```
y_pred = y_kmeans
y_pred = [3 if x==1 else x for x in y_kmeans]
y_pred = [1 if x==0 else x for x in y_pred]
y_pred = [0 if x==3 else x for x in y_pred]
print(y_pred)
```

```
[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, 1, 2, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1, 1, 2,
2, 2, 2, 1, 2, 1, 2, 1, 2, 2, 1, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2,
2, 1, 2, 2, 1]
```

Accuracy Score

In [34]:
```
from sklearn.metrics import accuracy_score
print('Accuracy of K means:',accuracy_score(y_trans, y_pred))
```

```
Accuracy of K means: 0.8933333333333333
```

Compare the Effect of Different Parameters

In [58]:
```
def dokms(X,y_pred,n_clrs=2,random_states=56,algorithms='full'):
    k = KMeans(n_clusters=n_clrs,random_state=random_states,algorithm=algorithms)
    y_km = k.fit_predict(X)
    accu1=accuracy_score(y_km,y_pred)
    return accu1
```

In [59]:
```
dokms(X,y_pred)
```

Out[59]: 0.7266666666666667

In [60]:
```
df=pd.DataFrame(columns=['Random State','No. of Clusters','Algorithms','Accuracy'])
df
```

Out[60]:

Random State	No. of Clusters	Algorithms	Accuracy

In [61]:
```
random_states=[9, 25, 45, 56]
ncl=[3,2,7]
alg=['auto','full','elkan']
```

In [62]:
```
for r_state in random_states:
    for nc in ncl:
        for a in alg:

            acc1 = dokms(X, y_pred, nc, r_state, a)

            dict1 = {}
            dict1['Random State'] = r_state
            dict1['Accuracy'] = acc1
            dict1['No. of Clusters'] = nc
            dict1['Algorithms'] = a

            df = df.append(dict1, ignore_index = True)
```

In [63]:
```
df.head()
```

Out[63]:

	Random State	No. of Clusters	Algorithms	Accuracy
0	9	3	auto	1.00
1	9	3	full	1.00
2	9	3	elkan	1.00
3	9	2	auto	0.02
4	9	2	full	0.02

In [64]:
```
df.tail()
```

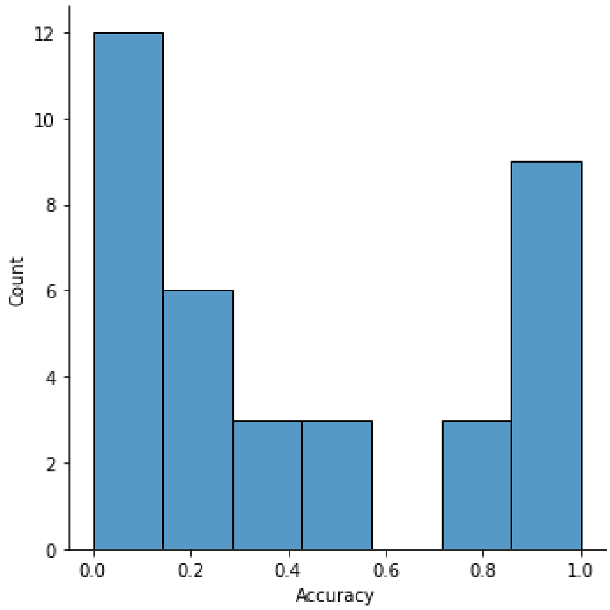
Out[64]:

	Random State	No. of Clusters	Algorithms	Accuracy
31	56	2	full	0.726667
32	56	2	elkan	0.726667
33	56	7	auto	0.000000
34	56	7	full	0.000000
35	56	7	elkan	0.000000

Visualization

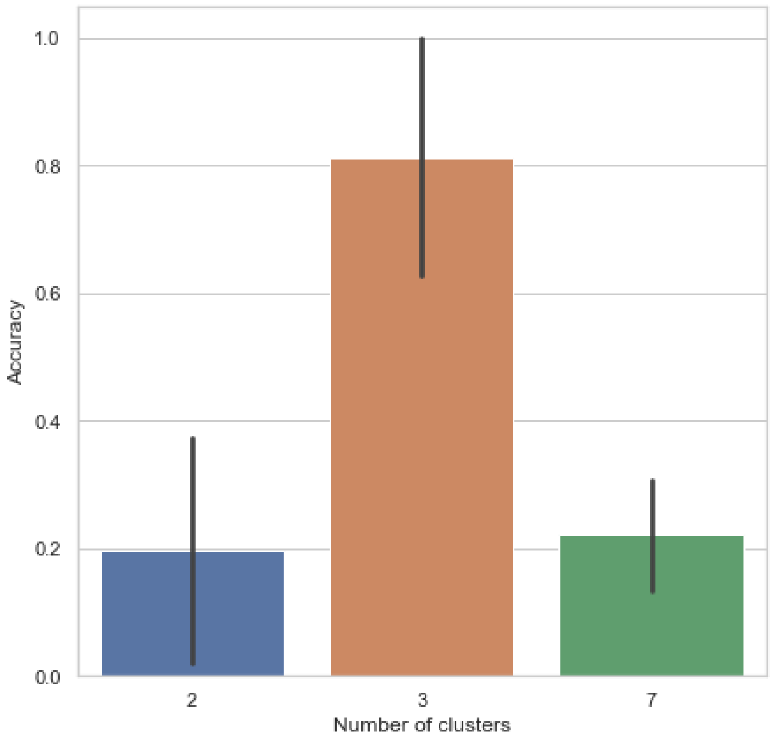
In [65]:
```
sns.displot(x = 'Accuracy', data = df)
```

Out[65]: <seaborn.axisgrid.FacetGrid at 0x23bdc6a3cd0>



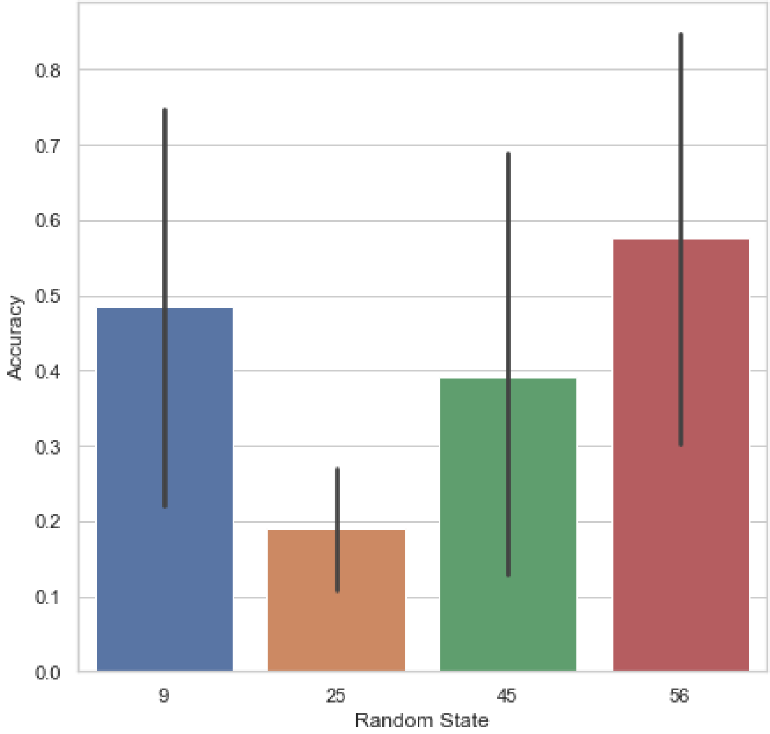
In [66]:
```
plt.figure(figsize = (7,7))
sns.set(style = "whitegrid")
f = sns.barplot(x = "No. of Clusters", y = "Accuracy", data = df)
f.set_xlabel("Number of clusters")
f.set_ylabel("Accuracy")
```

Out[66]: Text(0, 0.5, 'Accuracy')



In [71]:
```
plt.figure(figsize = (7,7))
sns.set(style = "whitegrid")
f = sns.barplot(x = "Random State", y = "Accuracy", data = df)
f.set_xlabel("Random State")
f.set_ylabel("Accuracy")
```

Out[71]: Text(0, 0.5, 'Accuracy')



Observation

-Accuracy range from 0 to 100 per.

-n_cluster=3 the accuracy value is high compared to other two values.

-Random state=56 have the highest accuracy and for 25 have the lowest.

Conclusion

In this lab, we have tried to implement K Means Clustering on 'Iris dataset'. Based on the observations, we are able to check the effect of clustering with respect to change in dataset split, random-state, hyperparameters. Also, we performed EDA with chosen dataset.