

Task3:IRIS FLOWER CLASSIFICATION

1.The Iris flower dataset consists of three species: setosa, versicolor, and virginica. These species can be distinguished based on their measurements. Now, imagine that you have the measurements of Iris flowers categorized by their respective species. Your objective is to train a machine learning model that can learn from these measurements and accurately classify the Iris flowers into their respective species.

2.Use the Iris dataset to develop a model that can classify iris flowers into different species based on their sepal and petal measurements. This dataset is widely used for introductory classification tasks.

1. Import Necessary Libraries

```
In [53]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

2. Load the Iris Dataset

```
In [54]: Iris_data=pd.read_csv('IRIS.csv')
Iris_data
```

Out[54]:

	sepal_length	sepal_width	petal_length	petal_width	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 × 5 columns

```
In [55]: #more detailed summary about the dataset such as no of records,columns names,  
#datatypes of columns and total computer memory consumed  
Iris_data.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    150 non-null    float64  
  1   sepal_width     150 non-null    float64  
  2   petal_length    150 non-null    float64  
  3   petal_width     150 non-null    float64  
  4   species         150 non-null    object  
dtypes: float64(4), object(1)  
memory usage: 6.0+ KB
```

```
In [56]: Iris_data.describe(include="all")
```

Out[56]:

	sepal_length	sepal_width	petal_length	petal_width	species
count	150.000000	150.000000	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	Iris-setosa
freq	NaN	NaN	NaN	NaN	50
mean	5.843333	3.054000	3.758667	1.198667	NaN
std	0.828066	0.433594	1.764420	0.763161	NaN
min	4.300000	2.000000	1.000000	0.100000	NaN
25%	5.100000	2.800000	1.600000	0.300000	NaN
50%	5.800000	3.000000	4.350000	1.300000	NaN
75%	6.400000	3.300000	5.100000	1.800000	NaN
max	7.900000	4.400000	6.900000	2.500000	NaN

```
In [57]: # checks for missing values in each column of the Iris dataset using the
#isnull() method and then sums up the number of missing values for each column using the sum() method
Iris_data.isnull().sum()
```

Out[57]:

sepal_length	0
sepal_width	0
petal_length	0
petal_width	0
species	0

dtype: int64

```
In [58]: #checking for different type of classes  
Iris_data["species"].value_counts()
```

```
Out[58]: Iris-setosa      50  
Iris-versicolor    50  
Iris-virginica     50  
Name: species, dtype: int64
```

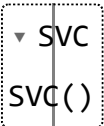
```
In [59]: X=Iris_data.drop('species',axis=1)  
y=Iris_data['species']
```

3. Split the Data into Training and Testing Sets

```
In [60]: ## Split the data into training and testing sets  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
In [61]: svm_model = SVC()
```

```
In [62]: svm_model.fit(X_train, y_train)
```

```
Out[62]: 
```

```
In [63]: y_pred = svm_model.predict(X_test)  
accuracy = accuracy_score(y_test, y_pred)  
print("Accuracy:", accuracy)
```

Accuracy: 1.0

```
In [64]: print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

Classification Report:

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	10
Iris-versicolor	1.00	1.00	1.00	9
Iris-virginica	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

```
In [65]: print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

Confusion Matrix:

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

```
In [ ]:
```