7/12

Aim: To convert categorical variables in a dataset in a dataset in a csv file into numerical data using python.

Program:

```
import pandas as pd
import numpy as np
df = pd.read_csv('empdata.csv')
df
```

	Sid	sage	gender	grade	employee
0	1	21	Male	1st class	yes
1	2	20	Female	2nd class	yes
2	3	22	Female	3rd class	no
3	4	21	Male	2nd class	yes
4	5	20	Male	1st class	no
5	6	19	Male	1st class	yes
6	7	18	Female	2nd class	no

```
df_categorical = df.select_dtypes(exclude = [np.number])
df_categorical
```

	gender	grade	employee
0	Male	1st class	yes
1	Female	2nd class	yes
2	Female	3rd class	no
3	Male	2nd class	yes
4	Male	1st class	no
5	Male	1st class	yes
6	Female	2nd class	no

→ df_categorical['grade'].unique()

out: array(['1st class', '2nd class', '3rd class'], dtype=object)

→ df_categorical.grade.value_counts()

out: grade
1st class 3
2nd class 3
3rd class 1
Name: count, dtype: int64

→ df_categorical.gender.value_counts()

out: gender
Male 4
Female 3
Name: count, dtype: int64

→ df_categorical.gender.replace({"Male":0, "Female":1, "mal":0}, inplace=True)
df_categorical

out:

	gender	grade	employee
0	0	1st class	yes
1	1	2nd class	yes
2	1	3rd class	no
3	0	2nd class	yes
4	0	1st class	no
5	0	1st class	yes
6	1	2nd class	no

→ df_categorical.employee.replace({"yes":0, "no":1}, inplace=True)
df_categorical.

out:

	gender	grade	employee
0	0	1st class	0
1	1	2nd class	0
2	1	3rd class	1
3	0	2nd class	0
4	0	1st class	1
5	0	1st class	0
6	1	2nd class	1

→ df_categorical.grade.replace({"1st class":1,"2nd class":2, "3rd class":3}, inplace=True)

df_categorical

out:

	gender	grade	employee
0	0	1	0
1	1	2	0
2	1	3	1
3	0	2	0
4	0	1	1
5	0	1	0
6	1	2	1

→ df_categorical.to_csv('new_file.csv')

da= pd.read_csv('new_file.csv')

da

	unnamed:0	gender	grade	employee
0	0	0	1	0
1	1	1	2	0
2	2	1	3	1
3	3	0	2	0
4	4	0	1	1
5	5	0	1	0
6	6	1	2	1

14/12

→ Aim: Decision Tree using Weka

Steps:

1) Open Weka tool & click on explorer
2) ~~Select classify and~~ click on open file, open weather.nominal
3) Select classify
4) Choose J48 classifier
5) Click on Start.

Output: (Use Training Set)

Detailed Accuracy by class

	TP rate	FP rate	Precision	Recall	F Measure	MCC	class
	1.000	0.000	1.000	1.000	1.000	1.000	yes
	1.000	0.000	1.000	1.000	1.000	1.000	no
Weighted avg	1.000	0.000	1.000	1.000	1.000	1.000	

= Confusion matrix =

a	b	← classified as
9	0	a = yes
0	5	b = no

c drive
↓
Weka
↓
data
↓
wn

= = Summary = =

Correctly classified instances	14	100 %
Incorrectly classified instances	0	0 %
Kappa Statistic	1	
Mean absolute Error	0	
Root mean squared Error	0	
Relative absolute Error	0 %	
Root relative squared error	0 %	
Total No. of Instances	14	

Using cross-validation:

=== Summary ===

Correctly classified instances	7	50%
Incorrectly classified instances	7	50%
Kappa Statistic	-0.0426	
Mean absolute Error	0.4167	
Root mean squared Error	0.5984	
Relative absolute Error	87.5%	
Root Relative Squared Error	121.2987%	
Total number of instances	14	

=== Detailed accuracy By Class ===

	TP Rate	FP	Precision	Recall	FMeasure	MCC	Class
	0.556	0.600	0.625	0.556	0.588	-0.043	yes
	0.400	0.444	0.333	0.400	0.364	-0.043	no
weighted avg	0.500	0.544	0.521	0.500	0.508	-0.043	

=== Confusion matrix ===

a	b	<-- classified as
5	4	a = yes
3	2	b = no

Using Percentage split:

=== Summary ===

Correctly classified instances	2	40%
Incorrectly classified instances	3	60%
Kappa Statistic	-0.3636	
Mean absolute error	0.6	
Root mean squared error	0.7746	
Relative absolute error	126.9231%	

Root relative squared error 157.6801%

Total number of instances 5

== Detailed Accuracy By Class ==

	TP rate	FP	Precision	recall	Fmeasure	MMC	class
	0.667	1.000	0.500	0.667	0.571	-0.408	yes
	0.000	0.333	0.000	0.000	0.000	-0.408	no
weighted avg	0.400	0.733	0.300	0.400	0.343	-0.408	

=== Confusion matrix ===

a	b	<-- classified as
2	1	a = yes
2	0	b = no

21/12

Aim: Decision Tree using Python

Program:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from matplotlib import pyplot as plt
from sklearn import tree
df = pd.read_csv('Iris.csv')
df.head()
```

→

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	1	5.1	3.5	1.4	0.2
1	2	4.9	3.0	1.4	0.2
2	3	4.7	3.2	1.3	0.2
3	4	4.6	3.1	1.5	0.2
4	5	5.0	3.6	1.4	0.2

```
Attribute = df.drop(['Id', 'Species'], axis='columns')
target = df['Species']
target
```

→

```
0      Iris-setosa
1      Iris-setosa
...
148    Iris-virginica
149    Iris-virginica
Name: Species, Length: 150, dtype: object
```

```
Att_train, Att_test, target_train, target_train
        = train_test_split (Attribute, target, test_size=0.3)

model = DecisionTreeClassifier()
model.fit(Att_train, target_train)
```

o/p → DecisionTreeClassifier()

```
predc = model.predict (Att_test)
predc
```

o/p → array (['Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
...]

```
→ confusion_matrix (target_test, predc)
```

output: array ([[16,0,0],
[0,12,0],
[0,1,16]], dtype=int64)

```
→ accuracy_score (target_test, predc)
```

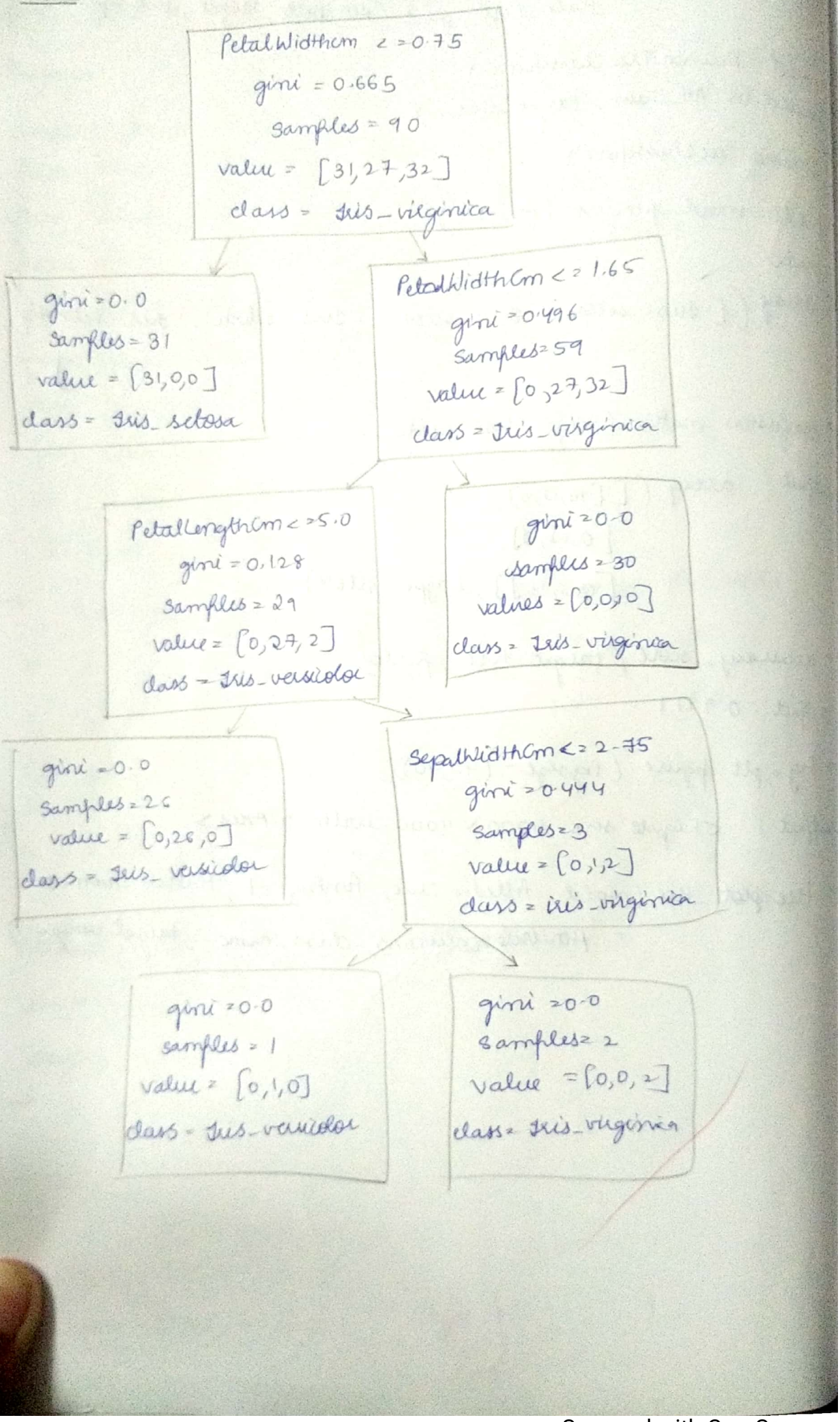
output: 0.9777

```
→ fig = plt.figure (figsize=(40,40))
```

output: <Figure size 4000 x 4000 with 0 Axes>

```
→ tree.plot_tree (model, filled=True, fontsize=7, feature_names
        = Attribute.columns, class_name = target.unique())
```

Output:



Aim: Bayesian classifier using Weka

Steps:

1) Open Weka, select Explorer
2) Open file, select weather.nominal dataset
3) Select classify and choose NaiveBayes classifier
4) For training dataset find classifier output

=== Summary ===

Correctly classified Instances	13	92.85%
Incorrectly classified Instances	1	7.1%
Kappa Statistic	0.8372	
Mean absolute Error	0.2917	
Root relative squared error	70.422%	
Total no of Instances	14	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	class
	1.000	0.200	0.900	1.000	yes
	0.800	0.000	1.000	0.800	no
Weighted Avg	0.929	0.129	0.936	0.929	

=== Confusion matrix ===

a	b	<-- classified as
9	0	a = yes
1	4	b = no

Aim: Program for Bayesian classification

Program:

```
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
from sklearn.metric import classification_report
from sklearn.metrics import accuracy_score
dataset = pd.read_csv('iris.csv')
Attribute = dataset.drop(['Id', 'Species'], axis = 'columns')
target = dataset['species']
x_train, x_test, y_train, y_test = train_test_split
                    (Attribute, target, test_size = 0.40)
classifier = GaussianNB()
classifier.fit(x_train, y_train)
```

O/P: GaussianNB()

```
→ y_pred = classifier.predict(x_test)
accuracy_score(y_test, y_pred)
```

O/P: 0.91666

```
→ confusion_matrix(y_test, y_pred)
```

O/P:

```
array([[20, 0, 0],
       [0, 17, 2],
       [0, 3, 18], dtype = int64)
```

→ cross_val_score (classifier, x_train, y_train, cv=5)

O/P: array ([0.9444, 1. , 1. , 1. , 0.9444])

→ cross_val_score (classifier, x_train, y_train, cv=3)

O/P: ([0.96666, 1. , 0.9666])

28/12

Aim: KNN algorithm implementation using weka.

Steps:

1) Open Weka, Select Explorer.
2) Select open files and open iris dataset.
3) Select classify
4) Choose from Lazy learner → IBK and give different K values.
5) Start and check for various values of K.

Output:

for K=1:

Correctly classified Instances	143	95.3333%
Incorrectly classified Instances	7	4.6667%

=== confusion matrix ===

a	b	c	<-- classified as
50	0	0	a = Iris_setosa
0	47	3	b = Iris_versicolor
0	4	46	c = Iris_virginica

for K=100

Correctly classified instances	100	66.6667%
Incorrectly classified instances	50	33.3333%

=== Confusion matrix ===

a	b	c	<-- classified as
50	0	0	a = Iris_setosa
0	50	0	b = Iris_versicolor
0	50	0	c = Iris_virginica

for K=10

correctly classified instances	144	96%
Incorrectly classified instances	6	4%

=== Confusion matrix ===

a	b	c	<-- classified as
50	0	0	a = Iris_setosa
0	48	2	b = Iris_versicolor
0	4	46	c = Iris_virginica

Aim: Write a python program to implement KNN.

Program:

```
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split,
                                    cross_val_score.

from sklearn.metrics import confusion_matrix,
                            classification_report,
                            accuracy_score

dataset = pd.read_csv ('Iris.csv')
dataset.head()
```

Output:

	Id	SepalLengthCm	SepalWidthCm	PetalLCm	PetalWcm	Species
0	1	5.1	3.5	1.4	0.2	Setosa
1	2	4.9	3.0	1.4	0.2	Setosa
2	3	4.7	3.2	1.3	0.2	Setosa
3	4	4.6	3.1	1.5	0.2	Setosa
4	5	5.0	3.6	1.4	0.2	Setosa

→ `attri = dataset.drop(['Species'], axis='columns')`

`attri`

Output:

Id	SLcm	SW	PL	PW
1	5.1	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
:	:	:	:	:
~~149~~	~~6~~	:	:	:
150	5.9	3.0	5.1	1.8

→ target = dataset ['Species']

target

output:

0 Setosa
1 Setosa
.
.
149 Virginica

Name: Species, Length: 150, dtype: object

→ attri_train, attri_test, target_train, target_test =
train_test_split (attri, target, test_size=0.5)

classifier = KNeighborsClassifier (n_neighbors=20)

classifier.fit (attri_train, target_train)

output: KNeighborsClassifier (n_neighbors=20)

→ pred = classifier.pred (attri_test)

pred

accuracy_score (target_test, pred)

output: 0.96

→ confusion_matrix (target_test, pred)

output: array ([[26, 0, 0],
[0, 25, 0],
[0, 3, 21]], dtype=int64)

→ import matplotlib.pyplot as plt

from pandas.plotting import scatter_matrix

features = attri.columns

features

output: Index (['Id', 'SLCm', 'SWCm', 'PLCm', 'PWCm'], dtype='object')

→ scatter_matrix(dataset [features])

graph

→ import plotly.express as px

fig = px.scatter_matrix(dataset, dimensions = attri.columns,
color = 'species')

fig.show()

29/12/23

11/1/24

Aim: KMeans algorithm using weka.

Steps:

1) In the preprocessing interface, open the weka explorer and load the required dataset, and we are taking the iris.arff dataset
2) Find the cluster tab and choose simple k-means algorithm
3) Then change the no.of clusters as per your wish
4) Click on start and select visualize cluster assignments to visualize the cluster.

Output:

=== Model and Evaluation on training set ===

Clustered Instances

0	19	(13%)
1	19	(13%)
2	50	(33%)
3	50	(33%)
4	12	(8%)

Cluster 0: 6.1, 2.9, 4.7, 1.4, Iris-versicolor

Cluster 1: 6.2, 2.9, 4.3, 1.3, Iris-versicolor

Cluster 2: 6.9, 3.1, 5.1, 2.3, Iris-Virginica

Cluster 3: 5.5, 4.2, 1.4, 0.2, Iris-setosa

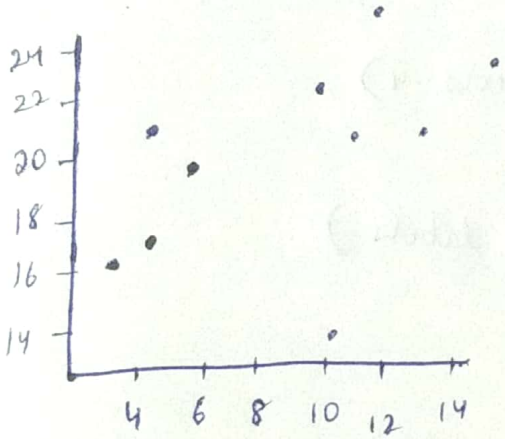
Cluster 4: 6.9, 3.1 H.9, 1.5, Iris-versicolor

Aim: Python program to implement KMeans algorithm

Program:

```
import numpy as np
import matplotlib.pyplot as plt
x = [4,5,10,4,3,11,14,8,10,12]
y = [21,19,14,17,16,25,24,22,21,21]
classes = [0,0,1,0,0,1,1,0,1,1]
plt.scatter(x,y,c=classes)
plt.show()
```

O/P:



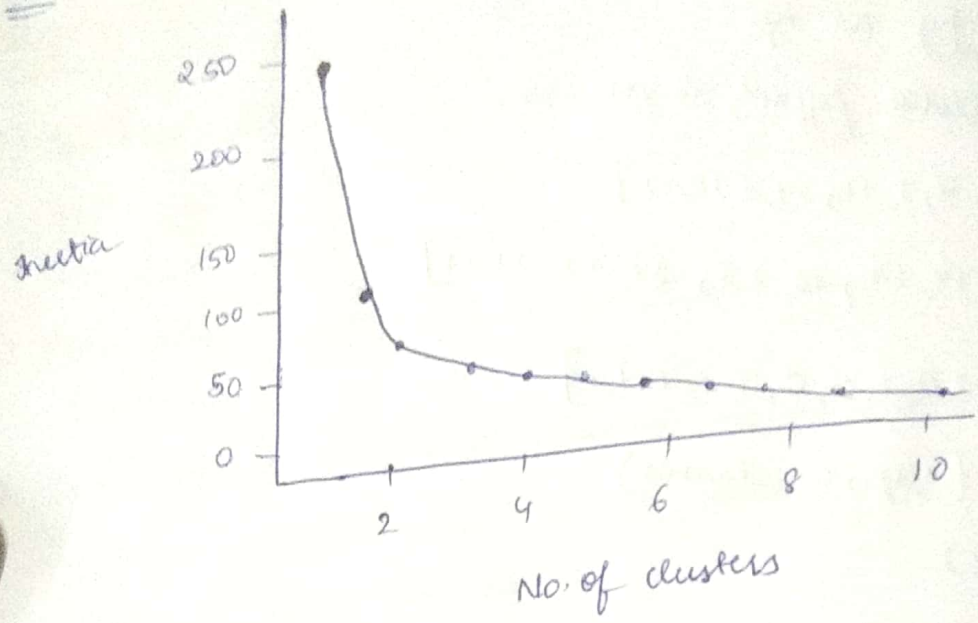
→

```
from sklearn.cluster import KMeans
data = list(zip(x,y))
inertias = []
for i in range(1,11):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(data)
    inertias.append(kmeans.inertia_)
plt.plot(range(1,11), inertias, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
```

plt.show()

Elbow Method

O/P :



→ kmeans = KMeans (n_clusters = 4)

kmeans.fit(data)

plt.scatter (x, y, c = kmeans.labels_)

plt.show()

O/P :