DWDM Laboratory Assignment Log Sheet 2025

Name: Bikash Thapa **Roll No: 28506/078**

Task No.	Title	Signature
1	Data Preprocessing and Cleaning	
2	Implementing Apriori Algorithm	
3	Implementing FP-growth	
4	Implementing Decision Tree (ID3) Algorithm	
5	Implementing Bayesian Classification Algorithm	
6	Implementing Support Vector Machine Classification Algorithm	
7	Implementing K-means Algorithm	
8	Implementing Mini-Batch K-Means Algorithm	
9	Implementing K-Medoids Algorithm	
10	Implementing Agglomerative Algorithm	
11	Implementing DBSCAN Algorithm	
12	Installing Data Mining Tool WEKA	

13	Data Visualization Using WEKA Explorer	

TASK 1: Data Preprocessing and Cleaning

Soure Code:

```
1.      print("Name: Bikash Thapa")
2.      print("Roll No: 28506/078")
3.      import pandas as pd
4.      import numpy as np
5.      from IPython.display import display_html   6. %matplotlib inline   7.
8.      def discretization(data):
9.      print("Discretizing 'Clump Thickness' attribute of the breast cancer dataset.\
10.     Visualizing distribution of attribute value")
11.     print(data['Clump Thickness'].value_counts(sort=False))  12.
13.     print("For the equal width method, we can apply the cut() function to discretize the
attribute\
14.     into 4 bins of similar interval widths.")
15.     print("The value_counts() function can be used to determine the number of instances in each
bin.")
16.     bins = pd.cut(data['Clump Thickness'], 4)  17.     print(bins.value_counts(sort=False))  18.
19.     print("For the equal frequency method, the qcut() function can be used to partition the\
20.     values into 4 bins such that each bin has nearly the same number of instances.")
21.     bins = pd.qcut(data['Clump Thickness'], 4)  22.     print(bins.value_counts(sort=False))  23.
24.     def sampling(data):
25.     print("Displaying the first five records of the table Without Sampling.")
26.     display_html(data.head())  27.
28.     print("A sample of size 3 is randomly selected (without
replacement) from the original data.")
29.     sample = data.sample(n=3)  30.     display_html(sample)
31.
32.     print("Randomly select 1% of the data (without replacement) and
display the selected samples.")
33.     sample = data.sample(frac=0.01, random_state=1)  34.
display_html(sample)  35.
36.     print("A sampling with replacement to create a sample whose size is equal to 1% of the entire
data.")
37.     sample = data.sample(frac=0.01, replace=True, random_state=1)
38.     display_html(sample)      39.
40.     def remove_duplicate(data):
41.     dups = data.duplicated()
42.     print('Number of duplicate rows = %d' % (dups.sum()))
43.     print('Number of rows before discarding duplicates = %d' % (data.shape[0]))
44.     data2 = data.drop_duplicates()
45.     print('Number of rows after discarding duplicates = %d' % (data2.shape[0]))  46.
47.     def outlier(data):
48.     data2 = data.drop(['Class'], axis=1)
49.     data2['Bare Nuclei'] = pd.to_numeric(data2['Bare Nuclei'])
50.     Z = (data2 - data2.mean()) / data2.std()
51.     print('Number of rows before discarding outliers = %d' % (Z.shape[0]))
52.     Z2 = Z.loc[((Z > -3).sum(axis=1) == 9) & ((Z <= 3).sum(axis=1) == 9), :]
53.     print('Number of rows after discarding missing values = %d' % (Z2.shape[0]))  54.
 55. def remove_missing(data):
```

```
56.     print('Number of rows in original data = %d' % (data.shape[0]))
```

```
    data = data.dropna()
    print('Number of rows after discarding missing values = %d' % (data.shape[0]))
    def replace_missing_value_by_median(data):
    data2 = pd.to_numeric(data['Bare Nuclei'], errors='coerce')
    print('Before replacing missing values:')
    print(data2[20:25])
    # Replace missing values with median
    data2 = data2.fillna(data2.median())
    print('\nAfter replacing missing values by median:')
    print(data2[20:25])
    def noise_handle(data):
    data = data.drop(['Sample code'], axis=1)
    data = data.replace('?', np.nan)   # ☑ FIXED np.NaN → np.nan

    print('Number of instances = %d' % (data.shape[0]))
    print('Number of attributes = %d' % (data.shape[1]))
    print('Number of missing values:')
        for col in data.columns:
        print('\t%s: %d' % (col, data[col].isna().sum()))
    print("To further preprocess select option:\
    0.Exit\
    1. Replace missing value by median\
    2. Remove missing value\
    3. Handle outlier\
    4. Remove duplicate\
    5. Sampling\
    6.Discretization:")
      option = int(input())
        while option != 0:
        if option == 1:
        replace_missing_value_by_median(data)
            elif option == 2:
            remove_missing(data)
            elif option == 3:
outlier(data)
        elif option == 4:
        remove_duplicate(data)
            elif option == 5:
            sampling(data)
            elif option == 6:
            discretization(data)
            else:
            print("Enter correct choice")
            print("Select your option again:")
            option = int(input())

    def view(data):
    print('Number of instances = %d' % (data.shape[0]))
    print('Number of attributes = %d' % (data.shape[1]))
    display_html(data.head())

        def main():
        data = pd.read_csv(
        'https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancerwisconsin/breast-cancer-wisconsin.data',
        header=None
        )
        data.columns = ['Sample code', 'Clump Thickness', 'Uniformity of Cell Size',
        'Uniformity of Cell Shape',
```

```
    'Marginal Adhesion', 'Single Epithelial Cell Size', 'Bare Nuclei', 'Bland Chromatin',
    'Normal Nucleoli', 'Mitoses', 'Class']

    print("Do you want to view data? (yes/no)")
    response = input()
    if response.lower() == 'yes': 123.         view(data)

     print("Do you want to remove noise and further preprocess data? (yes/no)") 126.
response = input()
     if response.lower() == 'yes': 128.
noise_handle(data)
     else: 130.
quit() 131.   132.
main() 133.
```

Input/Output

```
Name: Bikash Thapa
Roll No: 28506/078
```

```
Number of instances = 699
Number of attributes = 11
```

	Sample code	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
0	1000025	5	1	1	1	2	1	3	1	1	2
1	1002945	5	4	4	5	7	10	3	2	1	2
2	1015425	3	1	1	1	2	2	3	1	1	2
3	1016277	6	8	8	1	3	4	3	7	1	2
4	1017023	4	1	1	3	2	1	3	1	1	2

```
Do you want to remove noise and further preprocess data? (yes/no)
 yes
Number of instances = 699
Number of attributes = 10
Number of missing values:
        Clump Thickness: 0
        Uniformity of Cell Size: 0
        Uniformity of Cell Shape: 0
        Marginal Adhesion: 0
        Single Epithelial Cell Size: 0
        Bare Nuclei: 16
        Bland Chromatin: 0
        Normal Nucleoli: 0
        Mitoses: 0
        Class: 0
To further preprocess select option:    0.Exit    1. Replace missing value by median    2. Remove missing value    3. Handle outlier    4.
Remove duplicate    5. Sampling    6.Discretization:
 1
Before replacing missing values:
20    10.0
21     7.0
22     1.0
23     NaN
24     1.0
Name: Bare Nuclei, dtype: float64

After replacing missing values by median:
20    10.0
21     7.0
22     1.0
23     1.0
24     1.0
Name: Bare Nuclei, dtype: float64
Select your option again:
 2
Number of rows in original data = 699
Number of rows after discarding missing values = 683
Select your option again:
 3
Number of rows before discarding outliers = 699
Number of rows after discarding missing values = 632
Select your option again:
    4
   Number of duplicate rows = 236
   Number of rows before discarding duplicates = 699
   Number of rows after discarding duplicates = 463
   Select your option again:
    5
   Displaying the first five records of the table Without Sampling.
```

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
0	5	1	1	1	2	1	3	1	1	2
1	5	4	4	5	7	10	3	2	1	2
2	3	1	1	1	2	2	3	1	1	2
3	6	8	8	1	3	4	3	7	1	2
4	4	1	1	3	2	1	3	1	1	2

```
   A sample of size 3 is randomly selected (without replacement) from the original data.
```

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
26	3	2	1	1	1	1	2	1	1	2
227	8	9	9	5	3	5	7	7	1	4
415	3	3	2	6	3	3	3	5	1	2

```
Randomly select 1% of the data (without replacement) and display the selected samples.
```

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
584	5	1	1	6	3	1	1	1	1	2
417	1	1	1	1	2	1	2	1	1	2
606	4	1	1	2	2	1	1	1	1	2
349	4	2	3	5	3	8	7	6	1	4
134	3	1	1	1	3	1	2	1	1	2
502	4	1	1	2	2	1	2	1	1	2
117	4	5	5	10	4	10	7	5	8	4

```
A sampling with replacement to create a sample whose size is equal to 1% of the entire data.
```


```
A sampling with replacement to create a sample whose size is equal to 1% of the entire data.
```


	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
37	6	2	1	1	1	1	7	1	1	2
235	3	1	4	1	2	NaN	3	1	1	2
72	1	3	3	2	2	1	7	2	1	2
645	3	1	1	1	2	1	2	1	1	2
144	2	1	1	1	2	1	2	1	1	2
129	1	1	1	1	10	1	1	1	1	2
583	3	1	1	1	2	1	1	1	1	2

```
Select your option again:
 6
Discretizing 'Clump Thickness' attribute of the breast cancer dataset.    Visualizing distribution of attribute value
Clump Thickness
5     130
3     108
6      34
4      80
8      46
1     145
2      50
7      23
10     69
9      14
Name: count, dtype: int64
For the equal width method, we can apply the cut() function to discretize the attribute    into 4 bins of similar interval widths.
The value_counts() function can be used to determine the number of instances in each bin.
Clump Thickness
(0.991, 3.25]    303
(3.25, 5.5]      210
(5.5, 7.75]       57
(7.75, 10.0]     129
Name: count, dtype: int64
For the equal frequency method, the qcut() function can be used to partition the    values into 4 bins such that each bin has nearly the sa
me number of instances.
```

```
Select your option again:
 7
Enter correct choice
Select your option again:
 1
Before replacing missing values:
20    10.0
21     7.0
22     1.0
23     NaN
24     1.0
Name: Bare Nuclei, dtype: float64

After replacing missing values by median:
20    10.0
21     7.0
22     1.0
23     1.0
24     1.0
Name: Bare Nuclei, dtype: float64
```

TASK 2: Implementing Apriori Algorithm

Source Code:

```
print("Name: Bikash Thapa")
print("Roll No: 28506/078")
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
from IPython.display import display_html
def toy_dataset():
data = [
['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],
['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']
]
print("Do you want to view the raw data? (yes/no)")
choice = input()
if choice.lower() == 'yes':
print("Raw Data:")
print(data)
# One-hot encoding transaction data
te = TransactionEncoder()
te_ary = te.fit(data).transform(data)
df = pd.DataFrame(te_ary, columns=te.columns_)
print("Do you want to view the Encoded data? (yes/no)")
choice = input()
if choice.lower() == 'yes':
print("Encoded Data:")
display_html(df)
return df
def frequent_itemset(data):
print("Enter the value of minimum support threshold (e.g., 0.6):")
support = float(input())
frequent_itemsets = apriori(data, min_support=support, use_colnames=True)
print("Do you want to view frequent itemsets generated by Apriori? (yes/no)")
choice = input()
if choice.lower() == 'yes':
print("Frequent itemset:")
display_html(frequent_itemsets)
return frequent_itemsets
def association_rule(frequent_itemsets):
print("Enter your metric of interest (confidence/lift):")
choice = input().lower()
if choice == 'confidence':
print("Enter minimum confidence threshold value:")
min_confidence = float(input())
rule = association_rules(frequent_itemsets, metric="confidence",
min_threshold=min_confidence)
elif choice == 'lift':
print("Enter minimum lift threshold value:")
min_lift = float(input())
```

```
60.          rule = association_rules(frequent_itemsets, metric="lift", min_threshold=min_lift) 61.
else:
62.      print("Invalid choice. Defaulting to confidence with threshold 0.5")
63.      rule = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.5) 64.
65.      print("Do you want to view the learned association rules? (yes/no)")
66.      choice = input()
67.      if choice.lower() == 'yes':
68.      display_html(rule.drop(['leverage', 'conviction'], axis=1)) 69.      else:
70.          quit() 71.
72.      def main():
73.      data = toy_dataset()
74.      frequent_itemsets = frequent_itemset(data) 75.      association_rule(frequent_itemsets) 76.
77. main()
78.
```

Input/Output: Sample Rule 1

```
Name: Bikash Thapa
Roll No: 28506/078
Do you want to view the raw data? (yes/no) yes
Raw Data:
[['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'], ['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'], ['Milk', 'Apple', 'Kidney Beans', 'Eggs'], ['Milk', 'Unicorn', 'Corn', 'Ki
Do you want to view the Encoded data? (yes/no) yes
Encoded Data:
```

	Apple	Corn	Dill	Eggs	Ice cream	Kidney Beans	Milk	Nutmeg	Onion	Unicorn	Yogurt
0	False	False	False	True	False	True	True	True	True	False	True
1	False	False	True	True	False	True	False	True	True	False	True
2	True	False	False	True	False	True	True	False	False	False	False
3	False	True	False	False	False	True	True	False	False	True	True
4	False	True	False	True	True	True	False	False	True	False	False

```
Enter the value of minimum support threshold (e.g., 0.6):
 0.8
Do you want to view frequent itemsets generated by Apriori? (yes/no)
 yes
Frequent itemset:
```

	support	itemsets
0	0.8	(Eggs)
1	1.0	(Kidney Beans)
2	0.8	(Eggs, Kidney Beans)

```
Enter your metric of interest (confidence/lift):
 confidence
Enter minimum confidence threshold value:
 0.8
Do you want to view the learned association rules? (yes/no)
 yes
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	representativity	zhangs_metric	jaccard	certainty	kulczynski
0	(Eggs)	(Kidney Beans)	0.8	1.0	0.8	1.0	1.0	1.0	0.0	0.8	0.0	0.9
1	(Kidney Beans)	(Eggs)	1.0	0.8	0.8	0.8	1.0	1.0	0.0	0.8	0.0	0.9

Input/Output: Sample Rule 2

```
Name: Bikash Thapa
Roll No: 28506/078
Do you want to view the raw data? (yes/no) no
Do you want to view the Encoded data? (yes/no) no
```

```
Enter the value of minimum support threshold (e.g., 0.6):
 0.8
Do you want to view frequent itemsets generated by Apriori? (yes/no)
 no
Enter your metric of interest (confidence/lift):
 lift
Enter minimum lift threshold value:
 0.8
Do you want to view the learned association rules? (yes/no)
 yes
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	representativity	zhangs_metric	jaccard	certainty	kulczynski
0	(Kidney Beans)	(Eggs)	1.0	0.8	0.8	0.8	1.0	1.0	0.0	0.8	0.0	0.9
1	(Eggs)	(Kidney Beans)	0.8	1.0	0.8	1.0	1.0	1.0	0.0	0.8	0.0	0.9

TASK 3: Implementing FP-growth

Source Code:

```
1.      import pandas as pd
2.      from mlxtend.preprocessing import TransactionEncoder
3.      from mlxtend.frequent_patterns import fpgrowth
4.      from mlxtend.frequent_patterns import association_rules  5. from IPython.display import
display_html  6. def toy_dataset():
7.      data= [['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
8.      ['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
9.      ['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
10.     ['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],
11.     ['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']]
12.     print("Do you want to view the raw data?")
13.     choice =input() 14.     if choice=='yes':
15.     print("Raw Data:")
16.     print(data)
17.     te = TransactionEncoder()
18.     te_ary = te.fit(data).transform(data)
19.     df = pd.DataFrame(te_ary, columns=te.columns_)
20.     print("Do you want to view the Encoded data?")
21.     choice =input() 22.     if choice=='yes':
23.     print("Encoded Data:")
24.     display_html(df)  25.     return df 26.
27.     def frequent_itemset(data):
28.     print("Enter the value of minimum support threshold:")
29.     support=float(input())
30.     frequent_itemsets = fpgrowth(data, min_support=support, use_colnames=True)
31.     print("Do you want to view frequent itemsets generated by FP-growth?")
32.     choice =input() 33.     if choice=='yes':
34.     print("Frequent itemset:")
35.     display_html(frequent_itemsets) 36.     return
frequent_itemsets 37.
38.     def association_rule(frequent_itemsets):
39.     print("Enter your metric of interenst confidence or lift:")
40.     choice=input()
41.     if choice=='confidence':
42.     print("Enter minimum confidence threshold value:")
43.     min_conficence=float(input())
44.     rule=association_rules(frequent_itemsets, metric="confidence", min_threshold=min_conficence)
45.     elif choice=='lift':
46.     print("Enter minimum lift threshold value:")
47.     min_lift=float(input())
48.     rule=association_rules(frequent_itemsets, metric="lift", min_threshold=min_lift)
49.     print("Do you want to view the learned association rules?")
50.     choice=input() 51.     if choice=='yes':
52.         display_html(rule.drop(['leverage','conviction'],axis=1)) 53.
else:
54.         quit() 55.
56.     def main():
57.     data=toy_dataset()
58.     frequent_itemsets=frequent_itemset(data)
59.     association_rule(frequent_itemsets)
```

```
60. main() 61.
```

Input/Output: Sample Run 1

```
Name: Bikash Thapa
Roll No: 28506/078
Do you want to view the raw data? (yes/no) no
Do you want to view the Encoded data? (yes/no) no


Enter the value of minimum support threshold:
 0.6
Do you want to view frequent itemsets generated by FP-growth?
 no
Enter your metric of interenst confidence or lift:
 confidence
Enter minimum confidence threshold value:
 0.8
Do you want to view the learned association rules?
yes
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	representativity	zhangs_metric	jaccard	certainty	kulczynski
0	(Kidney Beans)	(Eggs)	1.0	0.8	0.8	0.8	1.00	1.0	0.0	0.80	0.0	0.900
1	(Eggs)	(Kidney Beans)	0.8	1.0	0.8	1.0	1.00	1.0	0.0	0.80	0.0	0.900
2	(Yogurt)	(Kidney Beans)	0.6	1.0	0.6	1.0	1.00	1.0	0.0	0.60	0.0	0.800
3	(Milk)	(Kidney Beans)	0.6	1.0	0.6	1.0	1.00	1.0	0.0	0.60	0.0	0.800
4	(Onion)	(Eggs)	0.6	0.8	0.6	1.0	1.25	1.0	0.5	0.75	1.0	0.875
5	(Onion)	(Kidney Beans)	0.6	1.0	0.6	1.0	1.00	1.0	0.0	0.60	0.0	0.800
6	(Onion, Eggs)	(Kidney Beans)	0.6	1.0	0.6	1.0	1.00	1.0	0.0	0.60	0.0	0.800
7	(Kidney Beans, Onion)	(Eggs)	0.6	0.8	0.6	1.0	1.25	1.0	0.5	0.75	1.0	0.875
8	(Onion)	(Kidney Beans, Eggs)	0.6	0.8	0.6	1.0	1.25	1.0	0.5	0.75	1.0	0.875

Input/Output: Sample Run 2

```
Name: Bikash Thapa
Roll No: 28506/078
Do you want to view the raw data? (yes/no) no
Do you want to view the Encoded data? (yes/no) no


Enter the value of minimum support threshold:
 0.6
Do you want to view frequent itemsets generated by FP-growth?
 no
Enter your metric of interenst confidence or lift:
 lift
Enter minimum lift threshold value:
 1.2
Do you want to view the learned association rules?
 yes
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	representativity	zhangs_metric	jaccard	certainty	kulczynski
0	(Eggs)	(Onion)	0.8	0.6	0.6	0.75	1.25	1.0	1.0	0.75	0.375	0.875
1	(Onion)	(Eggs)	0.6	0.8	0.6	1.00	1.25	1.0	0.5	0.75	1.000	0.875
2	(Kidney Beans, Eggs)	(Onion)	0.8	0.6	0.6	0.75	1.25	1.0	1.0	0.75	0.375	0.875
3	(Kidney Beans, Onion)	(Eggs)	0.6	0.8	0.6	1.00	1.25	1.0	0.5	0.75	1.000	0.875
4	(Eggs)	(Kidney Beans, Onion)	0.8	0.6	0.6	0.75	1.25	1.0	1.0	0.75	0.375	0.875
5	(Onion)	(Kidney Beans, Eggs)	0.6	0.8	0.6	1.00	1.25	1.0	0.5	0.75	1.000	0.875

TASK 4: Implementing Decision Tree (ID3) Algorithm

Source Code:

```
print("Name: Bikash Thapa")
print("Roll No: 28506/078")
import pandas as pd
from IPython.display import display_html
from sklearn import tree
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import subprocess   8.
def toy_dataset():
animal = [
['human',1,1,0,0,1,0,'mammals'],
['python',0,0,0,0,0,1,'reptiles'],
['salmon',0,0,1,0,0,0,'fishes'],
['whale',1,1,1,0,0,0,'mammals'],
['frog',0,0,1,0,1,1,'amphibians'],
['komodo',0,0,0,0,1,0,'reptiles'],
['bat',1,1,0,1,1,1,'mammals'],
['pigeon',1,0,0,1,1,0,'birds'],
['cat',1,1,0,0,1,0,'mammals'],
['leopard shark',0,1,1,0,0,0,'fishes'],
['turtle',0,0,1,0,1,0,'reptiles'],
['penguin',1,0,1,0,1,0,'birds'],
['porcupine',1,1,0,0,1,1,'mammals'],
['eel',0,0,1,0,0,0,'fishes'],
['salamander',0,0,1,0,1,1,'amphibians']
]
titles = ['Name','Warm_blooded','Give_birth','Aquatic_creature',
'Aerial_reature','Has_legs','Hibernates','Class']  29.
data = pd.DataFrame(animal, columns=titles)
data['Class'] = data['Class'].replace(
['fishes','birds','amphibians','reptiles'],'non-mammals'  33.     )
print("Do you want to view data? (yes/no)")
choice = input()
if choice.lower() == 'yes':
display_html(data)  38.     return data  39.
def build_model(data):
Y = data['Class']
X = data.drop(['Name','Class'], axis=1)
clf = tree.DecisionTreeClassifier(criterion='entropy', max_depth=3)
clf = clf.fit(X, Y)  45.     return clf  46.
def prediction_using_model(clf):
testData = [
['gila monster',0,0,0,0,1,1,'non-mammals'],
['platypus',1,0,0,0,1,1,'mammals'],
['owl',1,0,0,1,1,0,'non-mammals'],
['dolphin',1,1,1,0,0,0,'mammals']
]
titles = ['Name','Warm_blooded','Give_birth','Aquatic_creature',
'Aerial_reature','Has_legs','Hibernates','Class']
testData = pd.DataFrame(testData, columns=titles)  57.
print("Do you want to view test data? (yes/no)")
choice = input()
if choice.lower() == 'yes':
```

```
        display_html(testData)

    # Splitting test data
    y_test = testData['Class']
    x_test = testData.drop(['Name','Class'], axis=1)    y_pred = clf.predict(x_test)
    predictions = pd.concat(
    [testData['Name'], pd.Series(y_pred, name='Predicted Class')],
    axis=1
    )
    print("Prediction for your test data is:")    display_html(predictions)
    # Model evaluation
    print("Do you want to view Evaluation of model? (yes/no)")
    choice = input()
    if choice.lower() == 'yes':
    model_evaluation(y_pred, y_test)    else:
        quit()

    def model_evaluation(y_pred, y_test):
    print("Confusion Matrix:")
    report = (confusion_matrix(y_test, y_pred))
    cf = pd.DataFrame(report).transpose()    display_html(cf)
    score = accuracy_score(y_test, y_pred)
    print('Decision Tree Accuracy :', score)
    print("Classification report:")
    report = (classification_report(y_test, y_pred, output_dict=True))
    df = pd.DataFrame(report).transpose()
    display_html(df[['precision', 'recall','f1-score']].head(2))
    def main():
    data = toy_dataset()
    model = build_model(data)

    # export tree to dot format
    tree.export_graphviz(
    model, out_file="tree.dot",
    feature_names=data.drop(['Name','Class'], axis=1).columns,
    class_names=model.classes_,
    filled=True, rounded=True,
    special_characters=True     )
     # generate PNG using Graphviz
     subprocess.run(["dot", "-Tpng", "tree.dot", "-o", "tree.png"])

    print("Your decision tree constructed successfully, check the
current directory for tree.png")
    prediction_using_model(model)    main()
```

Sample Run 1:

```
Name: Bikash Thapa
Roll No: 28506/078
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
graphviz is already the newest version (2.42.2-6ubuntu0.1).
0 upgraded, 0 newly installed, 0 to remove and 35 not upgraded.
Do you want to view data? (yes/no)
yes
```

	Name	Warm_blooded	Give_birth	Aquatic_creature	Aerial_reature	Has_legs	Hibernates	Class
0	human	1	1	0	0	1	0	mammals
1	python	0	0	0	0	0	1	non-mammals
2	salmon	0	0	1	0	0	0	non-mammals
3	whale	1	1	1	0	0	0	mammals
4	frog	0	0	1	0	1	1	non-mammals
5	komodo	0	0	0	0	1	0	non-mammals
6	bat	1	1	0	1	1	1	mammals
7	pigeon	1	0	0	1	1	0	non-mammals
8	cat	1	1	0	0	1	0	mammals
9	leopard shark	0	1	1	0	0	0	non-mammals
10	turtle	0	0	1	0	1	0	non-mammals
11	penguin	1	0	1	0	1	0	non-mammals
12	porcupine	1	1	0	0	1	1	mammals
13	eel	0	0	1	0	0	0	non-mammals
14	salamander	0	0	1	0	1	1	non-mammals

```
Your decision tree constructed successfully, check the current directory for tree.png
Do you want to view test data? (yes/no)
 yes
```

	Name	Warm_blooded	Give_birth	Aquatic_creature	Aerial_reature	Has_legs	Hibernates	Class
0	gila monster	0	0	0	0	1	1	non-mammals
1	platypus	1	0	0	0	1	1	mammals
2	owl	1	0	0	1	1	0	non-mammals
3	dolphin	1	1	1	0	0	0	mammals

```
Prediction for your test data is:
```

	Name	Predicted Class
0	gila monster	non-mammals
1	platypus	non-mammals
2	owl	non-mammals
3	dolphin	mammals

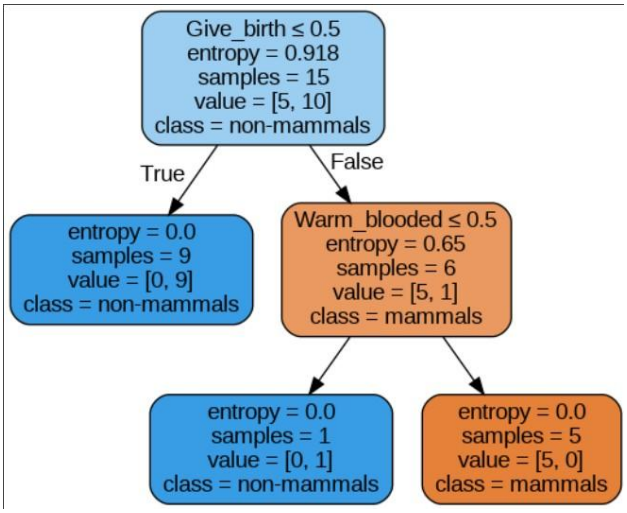
```
Do you want to view Evaluation of model? (yes/no)
 yes
Confusion Matrix:
```

	0	1
0	1	0
1	1	2

```
Decision Tree Accuracy : 0.75
Classification report:
```

	precision	recall	f1-score
mammals	1.000000	0.5	0.666667
non-mammals	0.666667	1.0	0.800000

If you open tree.png you will see the following decision tree:



Sample Run 2:

```
Name: Bikash Thapa
Roll No: 28506/078
Do you want to view data? (yes/no) no
Your decision tree constructed successfully, check the current directory for tree.png
Do you want to view test data? (yes/no) no
Prediction for your test data is:
```

	Name	Predicted Class
0	gila monster	non-mammals
1	platypus	non-mammals
2	owl	non-mammals
3	dolphin	mammals

TASK 5: Implementing Bayesian Classification Algorithm

Source Code:

```
1.      print("Name: Bikash Thapa")
2.      print("Roll No: 28506/078")
3.      import pandas as pd
4.      from IPython.display import display_html
5.      from sklearn.naive_bayes import GaussianNB
6.      from sklearn.model_selection import train_test_split
7.      from sklearn.metrics import accuracy_score, classification_report, confusion_matrix  8.
9.      def build_model(X_train, Y_train):
10.     clf = GaussianNB()
11.     clf = clf.fit(X_train, Y_train) 12.     return clf 13.
14.     def prediction_using_model(clf,X_test,Y_test):
15.     X_test = X_test.reset_index()
16.     del X_test['index']
17.     Y_test = Y_test.reset_index()
18.     del Y_test['index']
19.     Y_pred = clf.predict(X_test)
20.     predictions = pd.concat([X_test,pd.Series(Y_pred,name='Predicted Class')], axis=1)
21.     print("Do you want to view the class label prediction for top five tuples of test data?")
22.     choice=input() 23.     if choice=='yes':
24.     display_html(predictions.head())
25.     print("Do you want to view Evaluation result of model?")
26.     choice=input() 27.     if choice=='yes':
28.         model_evaluation(Y_pred,Y_test) 29.
else:
30.         quit() 31.
32.     def model_evaluation(y_pred,y_test):
33.     print("Confusion Matrix:")
34.     report=(confusion_matrix(y_test, y_pred))
35.     cf=pd.DataFrame(report).transpose()
36.     display_html(cf)
37.     score = accuracy_score(y_test,y_pred)
38.     print('Naive Bayes Accuracy :',score)
39.     print("Classification report:")
40.     report=(classification_report(y_test, y_pred, output_dict=True))
41.     df = pd.DataFrame(report).transpose()
42.     display_html(df[['precision', 'recall','f1-score']].head(3)) 43.
44.     def main():
45.     data = pd.read_csv('/Iris.csv')
46.     print("Do you want to view top five data tuples of Iris Dataset?")
47.     choice=input() 48.     if choice=='yes':
49.     display_html(data.head())
50.     Y = data['Species']
51.     X = data.drop(['Id','Species'],axis=1)
52.     X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=1)
53.      clf= build_model( X_train,Y_train)
54.      prediction_using_model(clf,X_test,Y_test)
55. main() 56.
```

Sample Run 1:

```
Name: Bikash Thapa
Roll No: 28506/078
Do you want to view top five data tuples of Iris Dataset?
yes
```

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

```
Do you want to view the class label prediction for top five tuples of test data?
yes
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Predicted Class
0	5.8	4.0	1.2	0.2	setosa
1	5.1	2.5	3.0	1.1	versicolor
2	6.6	3.0	4.4	1.4	versicolor
3	5.4	3.9	1.3	0.4	setosa
4	7.9	3.8	6.4	2.0	virginica

```
Do you want to view evaluation result of model?
yes
```

Confusion Matrix:

	0	1	2
0	13	0	0
1	0	15	1
2	0	0	9

Naive Bayes Accuracy : 0.9736842105263158

Classification report:

	precision	recall	f1-score
setosa	1.0	1.0000	1.000000
versicolor	1.0	0.9375	0.967742
virginica	0.9	1.0000	0.947368

Sample Run 2:

```
Name: Bikash Thapa
Roll No: 28506/078
Do you want to view top five data tuples of Iris Dataset?
no
Do you want to view the class label prediction for top five tuples of test data?
no
Do you want to view evaluation result of model?
yes
Confusion Matrix:
```

	0	1	2
0	13	0	0
1	0	15	1
2	0	0	9

```
Naive Bayes Accuracy : 0.9736842105263158
Classification report:
```

	precision	recall	f1-score
setosa	1.0	1.0000	1.000000
versicolor	1.0	0.9375	0.967742
virginica	0.9	1.0000	0.947368

TASK 6: Implementing Support Vector Machine Classification Algorithm

Source Code:

```
1.      print("Name: Bikash Thapa")
2.      print("Roll No: 28506/078")
3.      import pandas as pd
4.      from IPython.display import display_html
5.      from sklearn.svm import LinearSVC
6.      from sklearn.model_selection import train_test_split
7.      from sklearn.metrics import accuracy_score, classification_report, confusion_matrix  8.
9.      def build_model(X_train, Y_train):
10.     clf = LinearSVC()
11.     clf = clf.fit(X_train, Y_train) 12.     return clf 13.
14.     def prediction_using_model(clf,X_test,Y_test):
15.     X_test = X_test.reset_index()
16.     del X_test['index']
17.     Y_test = Y_test.reset_index()
18.     del Y_test['index']
19.     Y_pred = clf.predict(X_test)
20.     predictions = pd.concat([X_test,pd.Series(Y_pred,name='Predicted Class')], axis=1)
21.     print("Do you want to view the class label prediction for top five tuples of test data?")
22.     choice=input() 23.     if choice=='yes':
24.     display_html(predictions.head())
25.     print("Do you want to view Evaluation of model?")
26.     choice=input() 27.     if choice=='yes':
28.         model_evaluation(Y_pred,Y_test) 29.
else:
30.         quit() 31.
32.     def model_evaluation(y_pred,y_test):
33.     print("Confusion Matrix:")
34.     report=(confusion_matrix(y_test, y_pred))
35.     cf=pd.DataFrame(report).transpose()
36.     display_html(cf)
37.     score = accuracy_score(y_test,y_pred)
38.     print('SVM Accuracy :',score)
39.     print("Classification report:")
40.     report=(classification_report(y_test, y_pred, output_dict=True))
41.     df = pd.DataFrame(report).transpose()
42.     display_html(df[['precision', 'recall','f1-score']].head(3)) 43.
44.     def main():
45.     data = pd.read_csv(r"C:\Users\hp\Desktop\Bsc Csit 7th\DWDM\lab\Iris.csv")
46.     #Iris.csv file should be present in the current directory for this  download iris.csv data
file #form UCI machine learning repository
47.     print("Do you want to view to five data tuples of Iris Dataset?")
48.     choice=input() 49.     if choice=='yes':
50.     display_html(data.head())
51.     Y = data['Species']
52.     X = data.drop(['Id','Species'],axis=1)
53.     X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=1)
54.     clf= build_model( X_train,Y_train)
55.     prediction_using_model(clf,X_test,Y_test)
56. main() 57.
```

Sample Run 1:

```
Name: Bikash Thapa
Roll No: 28506/078
Do you want to view top five data tuples of Iris Dataset?
yes
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	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

```
Do you want to view the class label prediction for top five tuples of test data?
yes
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Predicted Class
0	5.8	4.0	1.2	0.2	setosa
1	5.1	2.5	3.0	1.1	versicolor
2	6.6	3.0	4.4	1.4	versicolor
3	5.4	3.9	1.3	0.4	setosa
4	7.9	3.8	6.4	2.0	virginica

```
Do you want to view Evaluation of model?
yes
Confusion Matrix:
```

	0	1	2
0	13	0	0
1	0	12	0
2	0	4	9

```
SVM Accuracy : 0.8947368421052632
```

Sample Run 2:

```
Classification report:
```

	precision	recall	f1-score
setosa	1.000000	1.00	1.000000
versicolor	1.000000	0.75	0.857143
virginica	0.692308	1.00	0.818182

TASK 7: Implementing K-means Algorithm

Source code:

```
1.      print("Name: Bikash Thapa")
2.      print("Roll No: 28506/078")
3.      import pandas as pd
4.      import numpy as np
5.      from IPython.display import display_html  6. from sklearn import cluster  7.
8.      def toy_dataset():
9.      ratings=[['Lokesh',5,5,2,1],['Jyoti',4,5,3,2],['Bijay',4,4,4,3],['Sita',2,2,4,5],['Manish',1,2,
3,4], ['Ram',2,1,5, 5]]
10.     titles = ['user','Loot','Chino','Ghar','Aatma']
11.     movies = pd.DataFrame(ratings,columns=titles)
12.     display_html(movies) 13.     return movies 14.
15.     def k_means_learn(k,movies):
16.     data = movies.drop('user',axis=1)
17.     k_means = cluster.KMeans(n_clusters=2, max_iter=50, random_state=1)
18.     k_means.fit(data)
19.     labels = k_means.labels_
20.     pd.DataFrame(labels, index=movies.user, columns=['Cluster ID'])
21.     print("Learned cluster centroids for two clusters o and 1:")
22.     centroids = k_means.cluster_centers_
23.     display_html(pd.DataFrame(centroids,columns=data.columns))
24.     print("Now You can use cluster centroids to other users to determine their cluster
assignments.")
25.     return(k_means) 26.
27.     def cluster_new_data(k_means, movies):
28.     testData = np.array([[4,5,1,2],[3,2,4,4],[2,3,4,1],[3,2,3,3],[5,4,1,4]])
29.     labels = k_means.predict(testData)
30.     labels = labels.reshape(-1,1)
31.     usernames = np.array(['Radhe','Riya','Pratik','Prativa','Shyam']).reshape(-1,1)
32.     cols = movies.columns.tolist()
33.     newusers = pd.DataFrame(np.concatenate((usernames, testData), axis=1),columns=cols)
34.     cols.append('Assigned Cluster')
35.     newusers_cluster = pd.DataFrame(np.concatenate((usernames, testData, labels), axis=1), columns
=cols)
36.     print("your New users (test data) are:")
37.     display_html(newusers)
38.     print("New Users with their assigned cluster:") 39.     display_html(newusers_cluster) 40.
41.     def main():
42.     k=2
43.     movies= toy_dataset()
44.     k_means=k_means_learn(k,movies)
45.     cluster_new_data(k_means, movies) 46. main() 47.
```

Input/Output

```
Name: Bikash Thapa
Roll No: 28506/078
```

	user	Loot	Chino	Ghar	Aatma
0	Lokesh	5	5	2	1
1	Jyoti	4	5	3	2
2	Bijay	4	4	4	3
3	Sita	2	2	4	5
4	Manish	1	2	3	4
5	Ram	2	1	5	5

```
Cluster assignment of training data:
```

user	Cluster ID
Lokesh	0
Jyoti	0
Bijay	0
Sita	1
Manish	1
Ram	1

	user	Loot	Chino	Ghar	Aatma
0	Radhe	4	5	1	2
1	Riya	3	2	4	4
2	Pratik	2	3	4	1
3	Prativa	3	2	3	3
4	Shyam	5	4	1	4

New Users with their assigned cluster:

	user	Loot	Chino	Ghar	Aatma	Assigned Cluster
0	Radhe	4	5	1	2	0
1	Riya	3	2	4	4	1
2	Pratik	2	3	4	1	0
3	Prativa	3	2	3	3	1
4	Shyam	5	4	1	4	0

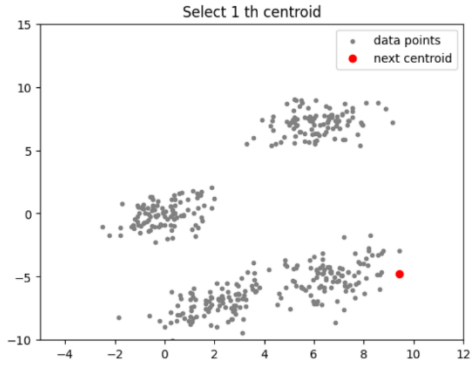
Source Code:

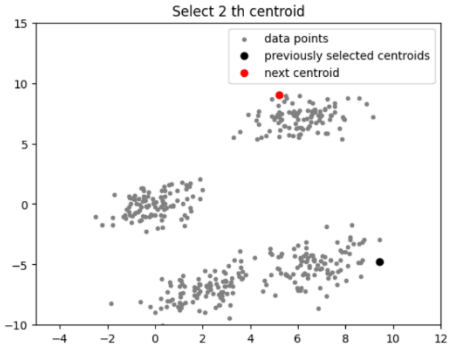
```
1.      print("Name: Bikash Thapa")
2.      print("Roll No: 28506/078")
3.      import numpy as np
4.      import pandas as pd
5.      import matplotlib.pyplot as plt
6.      %matplotlib inline  7. import sys  8.
9.      # First distribution
10.     mean_01 = np.array([0.0, 0.0])
11.     cov_01 = np.array([[1, 0.3], [0.3, 1]])
12.     dist_01 = np.random.multivariate_normal(mean_01, cov_01, 100) 13.
14.     # Second distribution
15.     mean_02 = np.array([6.0, 7.0])
16.     cov_02 = np.array([[1.5, 0.3], [0.3, 1]])
17.     dist_02 = np.random.multivariate_normal(mean_02, cov_02, 100) 18.
19.     # Third distribution
20.     mean_03 = np.array([7.0, -5.0])
21.     cov_03 = np.array([[1.2, 0.5], [0.5, 1.3]])
22.     dist_03 = np.random.multivariate_normal(mean_03, cov_03, 100)   23.
24.     # Fourth distribution
25.     mean_04 = np.array([2.0, -7.0])
26.     cov_04 = np.array([[1.2, 0.5], [0.5, 1.3]])
27.     dist_04 = np.random.multivariate_normal(mean_04, cov_04, 100) 28.
29.     # Combine all data
30.     data = np.vstack((dist_01, dist_02, dist_03, dist_04))
31.     np.random.shuffle(data) 32.
33. # Plot function
34. def plot(data, centroids):
```

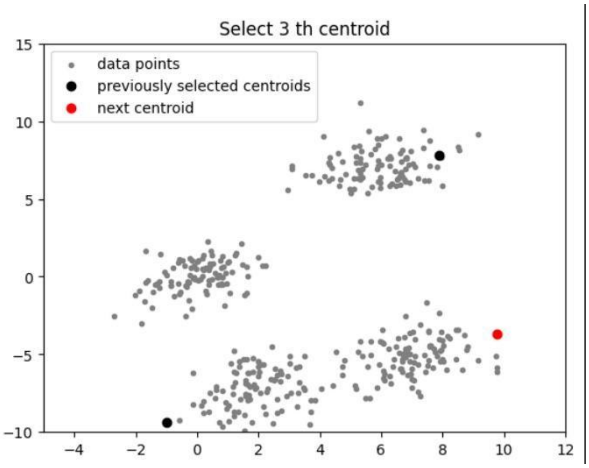
```
35.     plt.scatter(data[:, 0], data[:, 1], marker='.', color='gray', label='data points') 36.
if centroids.shape[0] > 1:
37.     plt.scatter(centroids[:-1, 0], centroids[:-1, 1], color='black', label='previously selected
centroids')
38.     plt.scatter(centroids[-1, 0], centroids[-1, 1], color='red', label='next centroid')
39.     plt.title('Select %d th centroid' % (centroids.shape[0]))
40.     plt.legend()
41.     plt.xlim(-5, 12)
42.     plt.ylim(-10, 15) 43.     plt.show() 44.
45. # Distance function 46.
def distance(p1, p2):
47.     return np.sum((p1 - p2)**2) 48.
49. # K-means++ Initialization 50.
def initialize(data, k):
51.     centroids = []
52.     centroids.append(data[np.random.randint(data.shape[0]), :])
53.     plot(data, np.array(centroids)) 54.
55.     for c_id in range(k - 1):
56.     dist = []
57.     for i in range(data.shape[0]):
58.     point = data[i, :]
59.     d = sys.maxsize
60.     for j in range(len(centroids)):
61.     temp_dist = distance(point, centroids[j])
62.     d = min(d, temp_dist) 63.             dist.append(d) 64.
65.     dist = np.array(dist)
66.     next_centroid = data[np.argmax(dist), :]
67.     centroids.append(next_centroid)
68.     plot(data, np.array(centroids)) 69.
70.     return np.array(centroids) 71.
72.     # Run initialization
73.     centroids = initialize(data, k=4) 74.
```

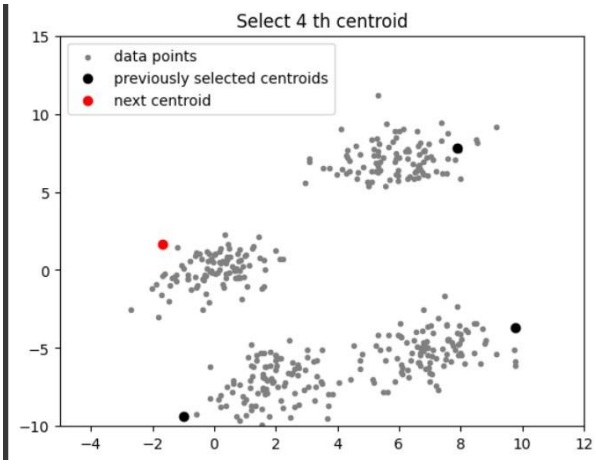
Input/Output

```
Name: Bikash Thapa
Roll No: 28506/078
```









TASK 8: Implementing Mini-Batch K-Means Algorithm

Source Code:

```
1.      print("Name: Bikash Thapa")
2.      print("Roll No: 28506/078")
3.      import matplotlib.pyplot as plt
4.      import pandas as pd
5.      import numpy as np
6.      from sklearn.cluster import MiniBatchKMeans
7.      from sklearn.metrics.pairwise import pairwise_distances_argmin
8.      from sklearn.datasets import make_blobs
9.      from IPython.display import display_html 10.
11. # Generate toy dataset 12.
def toy_dataset():
13.     centers = [[1, 1], [-2, -1], [1, -2], [1, 9]]
14.     n_clusters = len(centers)
15.     X, labels_true = make_blobs(n_samples=3000, centers=centers, cluster_std=0.9) 16.     titles
= ['x1', 'x2']
17.     data = pd.DataFrame(X, columns=titles)
18.     print("Do you want to view first 10 data elements? (yes/no)")
19.     choice = input()
20.     if choice.lower() == 'yes':
21.     display_html(data.head(10)) 22.     return data, X 23.
24. # Mini-batch k-means clustering 25.
def K_mini_batch(data):
26.     batch_size = 45
27.     mbk = MiniBatchKMeans(
28.     init='k-means++',
29.     n_clusters=4,
30.     batch_size=batch_size,
31.     n_init=10,
32.     max_no_improvement=10,
33.     verbose=0
34.     )
35.     mbk.fit(data)
36.     mbk_means_cluster_centers = np.sort(mbk.cluster_centers_, axis=0)
37.     labels = pairwise_distances_argmin(data, mbk_means_cluster_centers)
38.     print("Learned cluster centroids for three clusters:")
39.     centroids = mbk.cluster_centers_
40.     display_html(pd.DataFrame(centroids, columns=data.columns))
41.     return mbk, labels 42.
43. # Assign clusters to new data 44.
def cluster_new_data(mbk):
45.     testData = np.array([
46.     [-1.6434, 2.4534], [3.23423, -1.435], [1.345, 10.4390],
47.     [1.4345, 1.23454], [0.2311, -2.46345], [0.1675, -3345],
48.     [2.345323, 3.452234], [-2.234, -1.98823],
49.     [-1.7453, -1.934534], [1.23423, 2.4563]
50.     ])
51.     labels = mbk.predict(testData)
52.     labels = labels.reshape(-1, 1)
53.     cols = ['x1', 'x2']
54.     cols.append('Assigned Cluster')
55.     newdata_cluster = pd.DataFrame(np.concatenate((testData, labels), axis=1), columns=cols)
56.     display_html(newdata_cluster) 57.
58.     # Visualize clusters
59.     def view_cluster(labels, X, mbk):
60.     unique_labels = set(labels)
```

```
61.     colors = [plt.cm.Spectral(each) for each in np.linspace(0, 1, len(unique_labels))] 62.
63.     for k, col in zip(unique_labels, colors):
64.     class_member_mask = labels == k
65.     xy = X[class_member_mask]
66.     plt.plot(
67.     xy[:, 0], xy[:, 1], "o",
68.     markerfacecolor=tuple(col),
69.     markeredgecolor="k",
70.     markersize=6 71.          ) 72.
73.     # Plot centroids separately
74.     plt.plot(
75.     mbk.cluster_centers_[:, 0], mbk.cluster_centers_[:, 1], "o",
76.     markerfacecolor="cyan",
77.     markeredgecolor="k",
78.     markersize=12
79.     )
80.     plt.title("Mini-batch K-means clustering (Centroids in cyan)")
81.     plt.show() 82.
83. # Main function 84.
def main():
85.     data, X = toy_dataset()
86.     mbk, labels = K_mini_batch(data)
87.     print("Scatter plot of learned clusters:") 88.     view_cluster(labels, X, mbk) 89.
90.     print("Do you want clustering for new data based on learned clusters? (yes/no)")
91.     choice = input()
92.     if choice.lower() == 'yes': 93.         cluster_new_data(mbk) 94.     else:
95.         quit() 96.
97. main()
98.
```

Input/Output:

```
Name: Bikash Thapa
Roll No: 28506/078
Do you want to view first 10 data elements? (yes/
yes
```

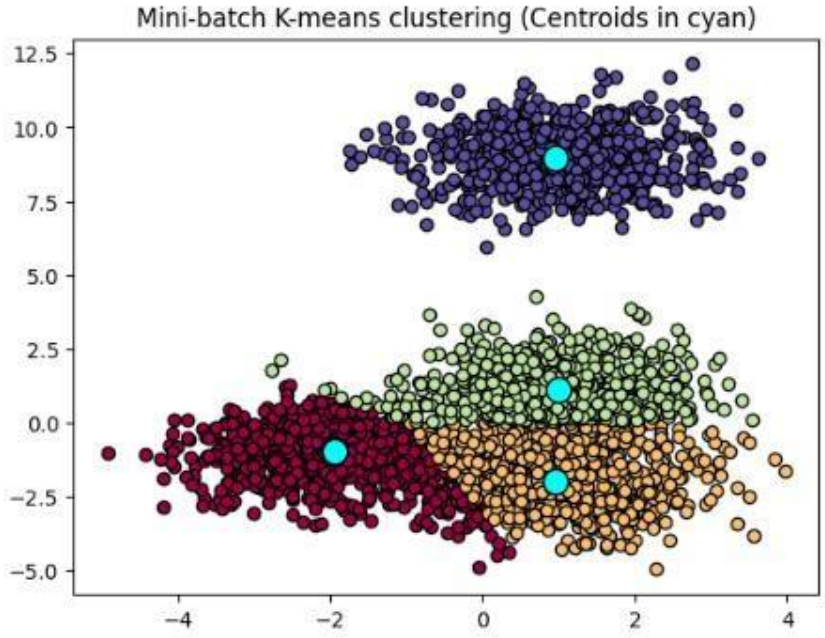
	x1	x2
0	0.315690	8.577669
1	-0.372466	-0.692144
2	-3.031504	-1.246741
3	1.700785	1.037400
4	-2.787623	-0.803647
5	2.021923	-1.512780
6	0.845539	7.937215
7	-0.677467	10.412902
8	2.161787	-1.742491
9	-0.349605	9.968550

```
Learned cluster centroids:
```

	x1	x2
0	0.966101	-1.995978
1	1.032477	8.913804
2	0.970435	0.977448
3	-2.004402	-0.964441

```
Scatter plot of learned clusters:
```

Scatter plot of learned clusters:



Do you want clustering for new data based on learned clusters? (yes/no)
yes

	x1	x2	Assigned Cluster
0	-1.643400	2.453400	2.0
1	3.234230	-1.435000	1.0
2	1.345000	10.439000	0.0
3	1.434500	1.234540	2.0
4	0.231100	-2.463450	1.0
5	0.167500	-3345.000000	1.0
6	2.345323	3.452234	2.0
7	-2.234000	-1.988230	3.0
8	-1.745300	-1.934534	3.0
9	1.234230	2.456300	2.0

TASK 9: Implementing K-Medoids Algorithm

Source Code:

```
print("Name: Bikash Thapa")
print("Roll No: 28506/078")
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from IPython.display import display_html  8.
def toy_dataset():
centers = [[1, 1], [-1, -1], [1, -1]]
X, _ = make_blobs(n_samples=750, centers=centers, cluster_std=0.4, random_state=0)
data = pd.DataFrame(X, columns=['x1','x2'])
print("Do you want to view first 10 data elements? (yes/no)")
choice = input()
if choice.lower() == 'yes':
display_html(data.head(10)) 17.     return data, X 18.
def manhattan_distance(a, b):
return np.sum(np.abs(a - b)) 21.
def KMedoids_manual(X, n_clusters=3, max_iter=50):
# randomly choose initial medoids
m = X[np.random.choice(len(X), n_clusters, replace=False)]
medoids = m.copy() 26.
for it in range(max_iter):
# assign points to nearest medoid
labels = np.array([np.argmin([manhattan_distance(x, medoid) for medoid in medoids]) for x in
X]) 30.
# update medoids
new_medoids = medoids.copy() 33.         for k in
range(n_clusters):
            cluster_points = X[labels == k] 35.
if len(cluster_points) > 0:
distances = np.sum(np.abs(cluster_points[:, None] - cluster_points[None, :]),
axis=2)
new_medoid_idx = np.argmin(distances.sum(axis=1)) 38.
new_medoids[k] = cluster_points[new_medoid_idx] 39.         if np.all(new_medoids ==
medoids):
break
medoids = new_medoids 42.     return medoids, labels 43.
def predict_new_data(medoids, new_points):
labels = np.array([np.argmin([manhattan_distance(x, medoid) for medoid in medoids]) for x in
new_points])
labels = labels.reshape(-1,1)
cols = ['x1','x2']
cols.append('Assigned Cluster')
newdata_cluster = pd.DataFrame(np.concatenate((new_points, labels), axis=1), columns=cols)
display_html(newdata_cluster) 51.
def view_cluster(labels, X, medoids):
unique_labels = set(labels)
colors = [plt.cm.Spectral(each) for each in np.linspace(0, 1, len(unique_labels))] 55.
plt.figure(figsize=(8,6))
        for k, col in zip(unique_labels, colors):
        class_member_mask = labels == k
```

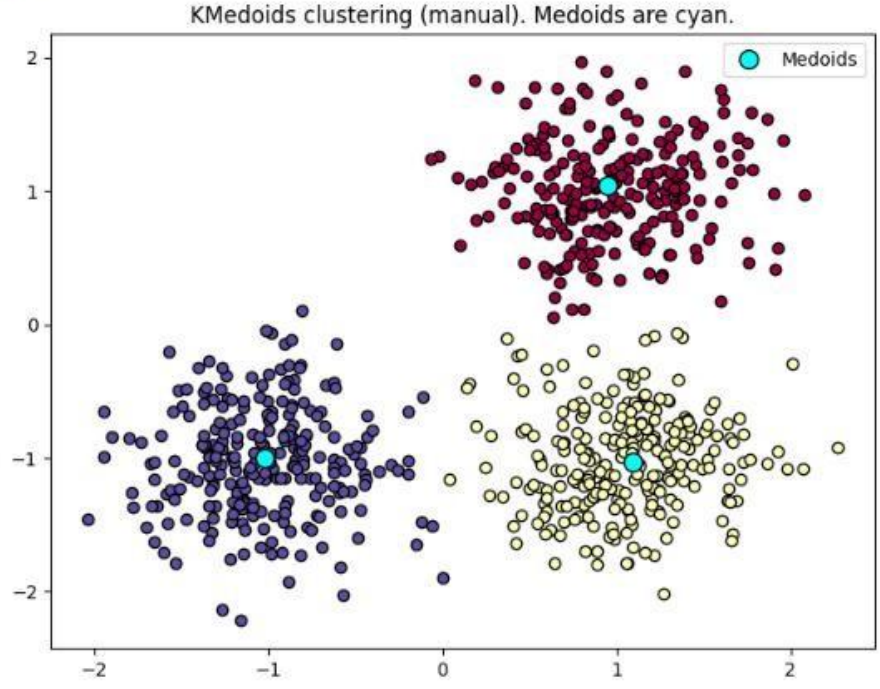
```
    xy = X[class_member_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
    markeredgecolor='k', markersize=6)
    plt.plot(medoids[:,0], medoids[:,1], 'o', markerfacecolor='cyan',
    markeredgecolor='k', markersize=10, label='Medoids')
    plt.title("KMedoids clustering (manual). Medoids are cyan.")
    plt.legend() 65.     plt.show() 66.
    def main():
    data, X = toy_dataset()
    X_np = data.values
    medoids, labels = KMedoids_manual(X_np, n_clusters=3, max_iter=50) 71.
     print("Do you want to view the scatter plot of learned clusters? (yes/no)") 73.
choice = input()
    if choice.lower() == 'yes':
    view_cluster(labels, X_np, medoids) 76.
    print("Do you want clustering for new data based on learned clusters? (yes/no)")
    choice = input()
    if choice.lower() == 'yes':
    testData = np.array([
    [0.81,1.12], [-1.145,-1.194], [0.676,0.7133], [0.4442,-1.3245],
    [1.23623,1.34634], [-0.93423,-0.0332], [-1.00234,-1.546],
    [0.946,-0.4674], [1.534,0.4789], [1.23523,1.0547]
    ])
    predict_new_data(medoids, testData) 86.
main() 88.
```

Input/Output: Sample Run 1

```
Name: Bikash Thapa
Roll No: 28506/078
Do you want to view first 10 data elements? (yes/no)
yes
```

	x1	x2
0	0.840220	1.148022
1	-1.154748	-1.204117
2	0.678636	0.724180
3	0.450783	-1.427097
4	1.492116	1.480952
5	-0.981946	-0.064150
6	-1.023435	-1.127017
7	0.935113	-0.692228
8	1.450654	0.568027
9	1.379768	1.035020

```
Do you want to view the scatter plot of learned clusters? (yes/no)
 yes
```



```
Do you want clustering for new data based on learned clusters? (yes/no)
 yes
```

	x1	x2	Assigned Cluster
0	0.81000	1.12000	0.0
1	-1.14500	-1.19400	2.0
2	0.67600	0.71330	0.0
3	0.44420	-1.32450	1.0
4	1.23623	1.34634	0.0
5	-0.93423	-0.03320	2.0
6	-1.00234	-1.54600	2.0
7	0.94600	-0.46740	1.0
8	1.53400	0.47890	0.0
9	1.23523	1.05470	0.0

TASK 10: Implementing Agglomerative Algorithm

Source Code:

```
print("Name: Bikash Thapa")
print("Roll No: 28506/078")
import pandas as pd
from IPython.display import display_html
from scipy.cluster import hierarchy
import matplotlib.pyplot as plt  7. %matplotlib inline
 8.
def toy_dataset():
animal = [
['human',1,1,0,0,1,0,'mammals'], ['python',0,0,0,0,0,1,'reptiles'],
['salmon',0,0,1,0,0,0,'fishes'], ['whale',1,1,1,0,0,0,'mammals'],
['frog',0,0,1,0,1,1,'amphibians'], ['komodo',0,0,0,0,1,0,'reptiles'],
['bat',1,1,0,1,1,1,'mammals'],['pigeon',1,0,0,1,1,0,'birds'],
['cat',1,1,0,0,1,0,'mammals'], ['leopard shark',0,1,1,0,0,0,'fishes'],
['turtle',0,0,1,0,1,0,'reptiles'], ['penguin',1,0,1,0,1,0,'birds'],
['porcupine',1,1,0,0,1,1,'mammals'], ['eel',0,0,1,0,0,0,'fishes'],
['salamander',0,0,1,0,1,1,'amphibians']
]
titles =
['Name','Warm_blooded','Give_birth','Aquatic_creature','Aerial_reature','Has_legs','Hibernates','Cla
ss']
data = pd.DataFrame(animal, columns=titles)
print("Do you want to view data? (yes/no)")
choice = input()
if choice.lower() == 'yes':
display_html(data) 26.     return data 27.
def ward(names, X):
Z = hierarchy.linkage(X.values, 'ward')
hierarchy.dendrogram(Z, labels=names.tolist(), orientation='right')
plt.show() 32.
def centroid(names, X):
Z = hierarchy.linkage(X.values, 'centroid')
hierarchy.dendrogram(Z, labels=names.tolist(), orientation='right')
plt.show() 37.
def group_average(names, X):
Z = hierarchy.linkage(X.values, 'average')
hierarchy.dendrogram(Z, labels=names.tolist(), orientation='right')
plt.show() 42.
def complete_link(names, X):
Z = hierarchy.linkage(X.values, 'complete')
hierarchy.dendrogram(Z, labels=names.tolist(), orientation='right')
plt.show() 47.
def single_link(names, X):
Z = hierarchy.linkage(X.values, 'single')
hierarchy.dendrogram(Z, labels=names.tolist(), orientation='right')
plt.show() 52.
def main():
data = toy_dataset()
names = data['Name']
X = data.drop(['Name','Class'], axis=1)
print("Your data is ready!")
```

```
58.     print("Select your option:\n1. single_Link\n2.
Complete_Link\n3. Group_average\n4.
Centroid\n5. Ward")
59.     choice = int(input()) 60.     if choice == 1:
61.         single_link(names, X) 62.
elif choice == 2:
63.     complete_link(names, X)
64.     elif choice == 3:
65.     group_average(names, X) 66.     elif choice == 4:
67.         centroid(names, X) 68.
elif choice == 5:
69.         ward(names, X) 70.
else:
71.         print("Enter correct choice next time")
72.         quit() 73.
74. main() 75.
```

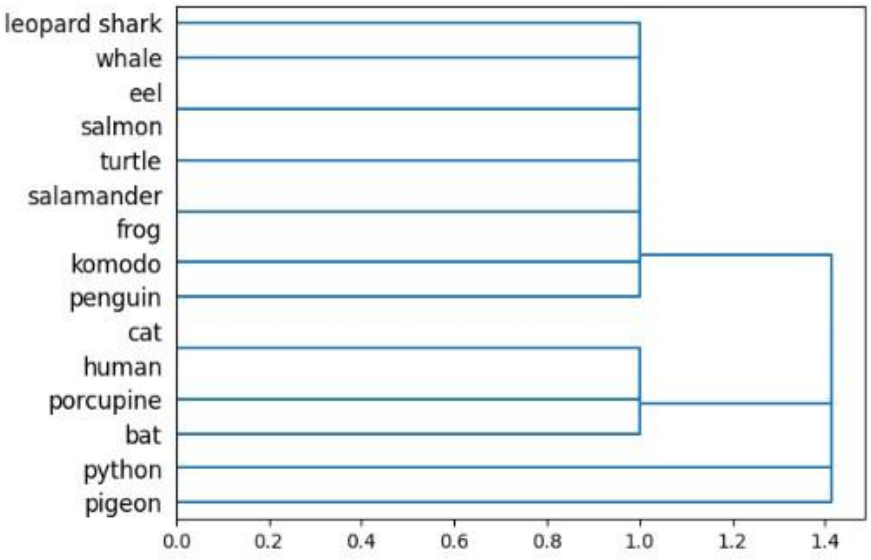
Input/Output: Sample Run 1

```
Name: Bikash Thapa
Roll No: 28506/078
Do you want to view data? (yes/no)
yes
```

	Name	Warm_blooded	Give_birth	Aquatic_creature	Aerial_creature	Has_legs	Hibernates	Class
0	human	1	1	0	0	1	0	mammals
1	python	0	0	0	0	0	1	reptiles
2	salmon	0	0	1	0	0	0	fishes
3	whale	1	1	1	0	0	0	mammals
4	frog	0	0	1	0	1	1	amphibians
5	komodo	0	0	0	0	1	0	reptiles
6	bat	1	1	0	1	1	1	mammals
7	pigeon	1	0	0	1	1	0	birds
8	cat	1	1	0	0	1	0	mammals
9	leopard shark	0	1	1	0	0	0	fishes
10	turtle	0	0	1	0	1	0	reptiles
11	penguin	1	0	1	0	1	0	birds
12	porcupine	1	1	0	0	1	1	mammals
13	eel	0	0	1	0	0	0	fishes
14	salamander	0	0	1	0	1	1	amphibians

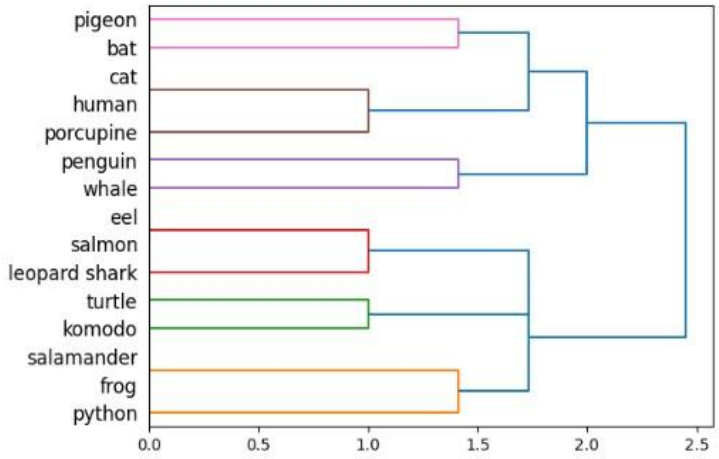
```
Your data is ready!
```

```
Your data is ready!
Select your option:
1. single_Link
2. Complete_Link
3. Group_average
4. Centroid
5. Ward
 1
```



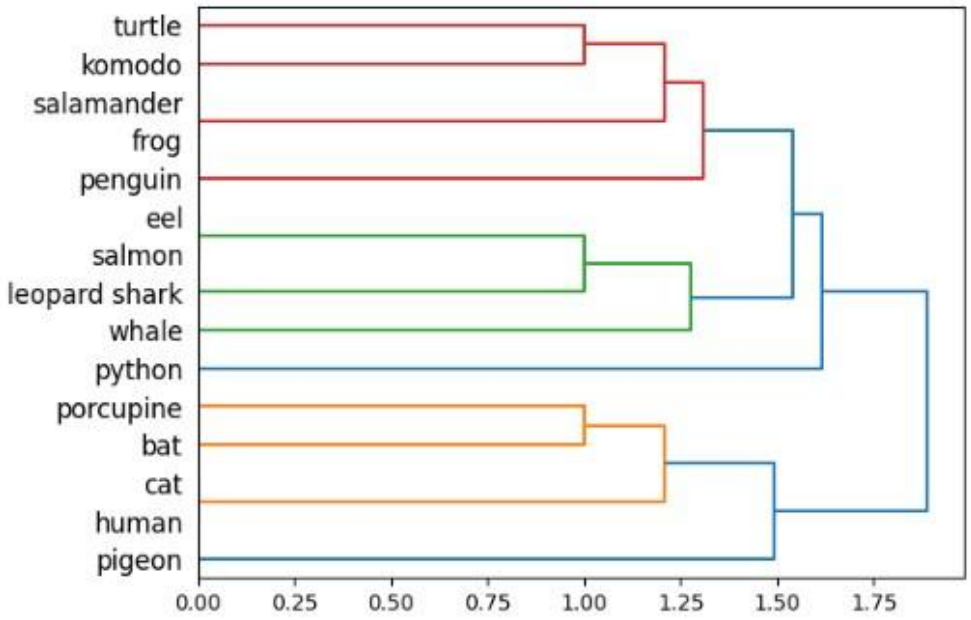
Input/Output: Sample run 2

```
Do you want to view data? (yes/no)
 no
Your data is ready!
Select your option:
1. single_Link
2. Complete_Link
3. Group_average
4. Centroid
5. Ward
 2
```



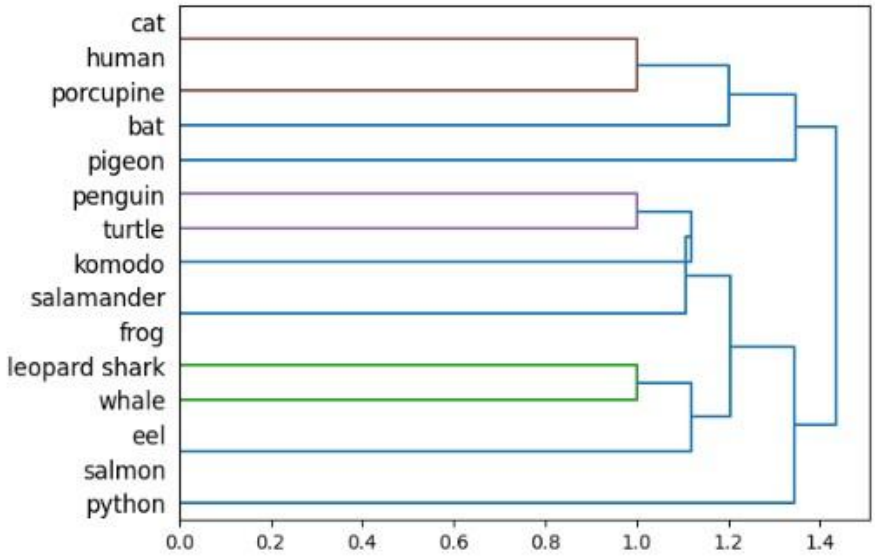
Input/Output: Sample run 3

```
Do you want to view data? (yes/no)
 no
Your data is ready!
Select your option:
1. single_Link
2. Complete_Link
3. Group_average
4. Centroid
5. Ward
 3
```



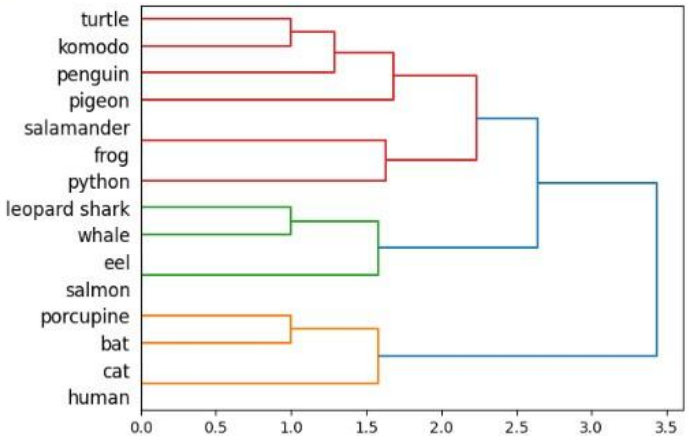
Input/Output: Sample run 4

```
Do you want to view data? (yes/no)
 no
Your data is ready!
Select your option:
1. single_Link
2. Complete_Link
3. Group_average
4. Centroid
5. Ward
 4
```



Input/Output: Sample run 5

```
Do you want to view data? (yes/no)
 no
Your data is ready!
Select your option:
1. single_Link
2. Complete_Link
3. Group_average
4. Centroid
5. Ward
 5
```



TASK 11: Implementing DBSCAN Algorithm

Source code :

```
1.      print("Name: Bikash Thapa")
2.      print("Roll No: 28506/078")
3.      from sklearn.cluster import DBSCAN
4.      import pandas as pd
5.      import numpy as np
6.      from IPython.display import display_html  7. def toy_dataset():
8.      value= np.array([[1, 3], [4, 2], [2, 3],[6, 7], [8, 9],[10,12],[10,15],
9.      [50, 19], [20, 20], [44, 21],[46, 18], [51, 19], 10.                  [50,
10], [40, 22], [50, 23],[49, 53], [50, 25], 11.                  [50, 50], [50, 45],
[44, 51],[46, 55], [51, 48],
12.     [50, 58], [40, 49], [50, 55],[49, 53], [50, 52],
13.     [25, 80], [100, 30],[150, 90]])
14.     titles = ['x','y']
15.     data = pd.DataFrame(value,columns=titles)
16.     print("Do you want to view data?")
17.     option=input() 18.     if option=='yes':
19.     print("First five data points:")
20.     display_html(data.head())
21.     print("Do you want to view scatter plot of data?")
22.     option=input() 23.     if option=='yes':
24.     print("Data points scatter plot:")
25.     data.plot.scatter(x='x',y='y') 26.     return data 27.
28.     def Dbscan_clustering(data):
29.     db = DBSCAN(eps=10.5, min_samples=4).fit(data)
30.     core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
31.     core_samples_mask[db.core_sample_indices_] = True
32.     labels = pd.DataFrame(db.labels_,columns=['Cluster ID'])
33.     result = pd.concat((data,labels), axis=1)
34.     result.plot.scatter(x='x',y='y',c='Cluster ID', colormap='jet') 35.
36.     def main():
37.     data=toy_dataset()
38.     print("Clusters contructed by DBSCAN:")
39.     Dbscan_clustering(data) 40. main() 41.
```

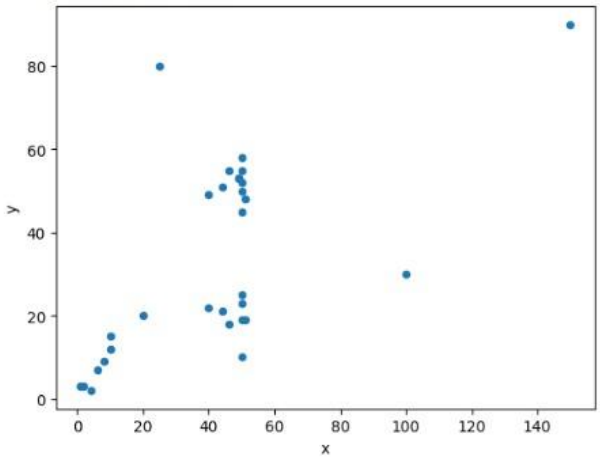
Input/Output

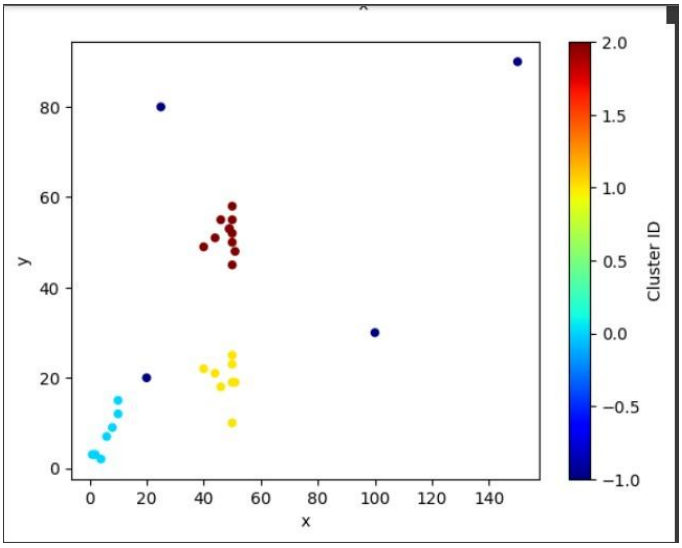
```
Name: Bikash Thapa
Roll No: 28506/078
Do you want to view data? (yes/no)
yes
First five data points:
```

	x	y
0	1	3
1	4	2
2	2	3
3	6	7
4	8	9

```
Do you want to view scatter plot of data? (yes/no)
```

```
Do you want to view scatter plot of data?
 yes
Data points scatter plot:
Clusters contructed by DBSCAN:
```



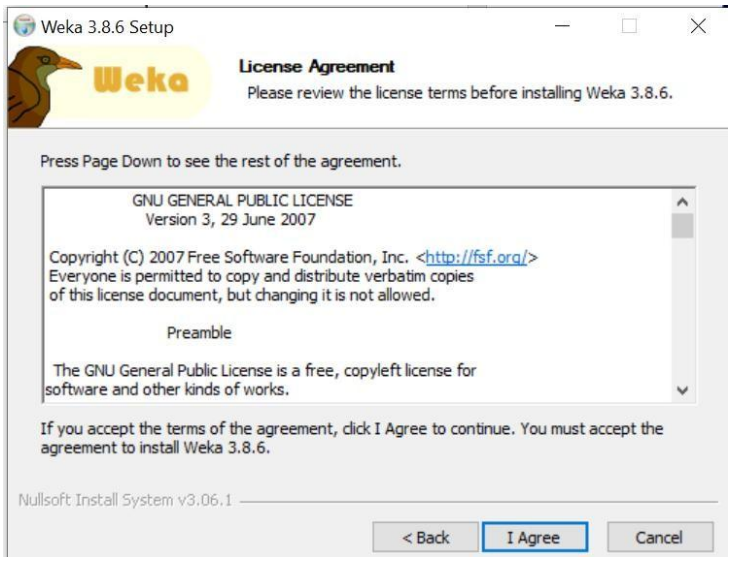


TASK 12: Installing Data Mining Tool WEKA

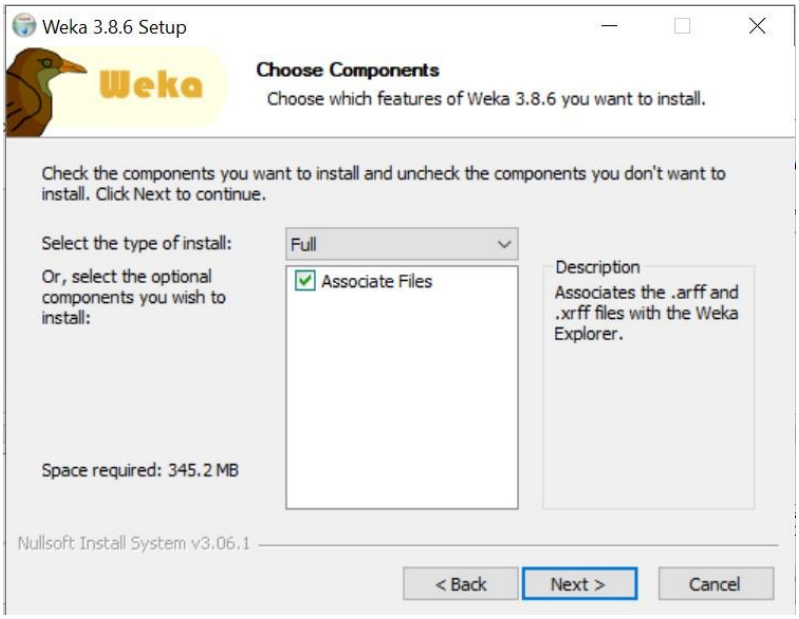
Following are the steps to install the WEKA.

1. Download WEKA software: https://waikato.github.io/weka-wiki/downloading_weka/
2. After successful download, open the file location and double click on the downloaded file. The StepUp wizard will appear. Click on Next.

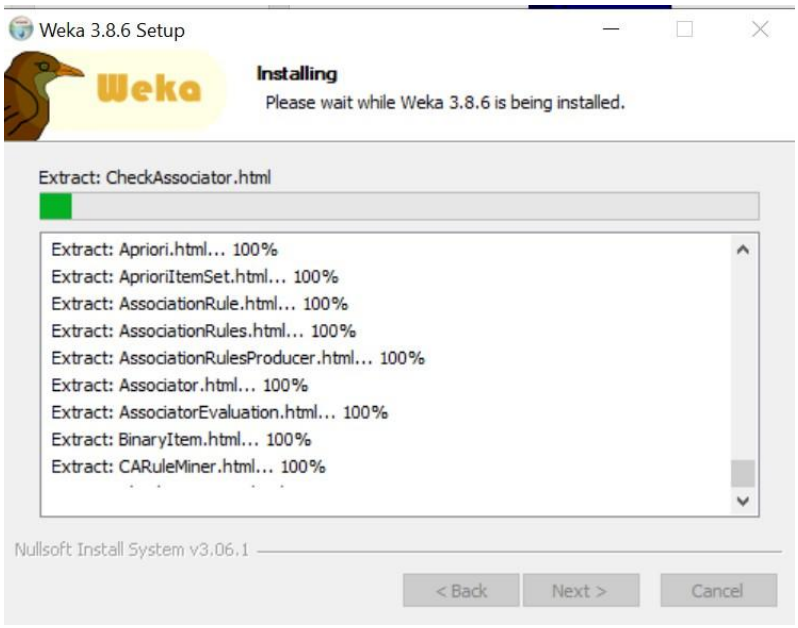
3. The License Agreement terms will open. Read it thoroughly and click on “I Agree”.



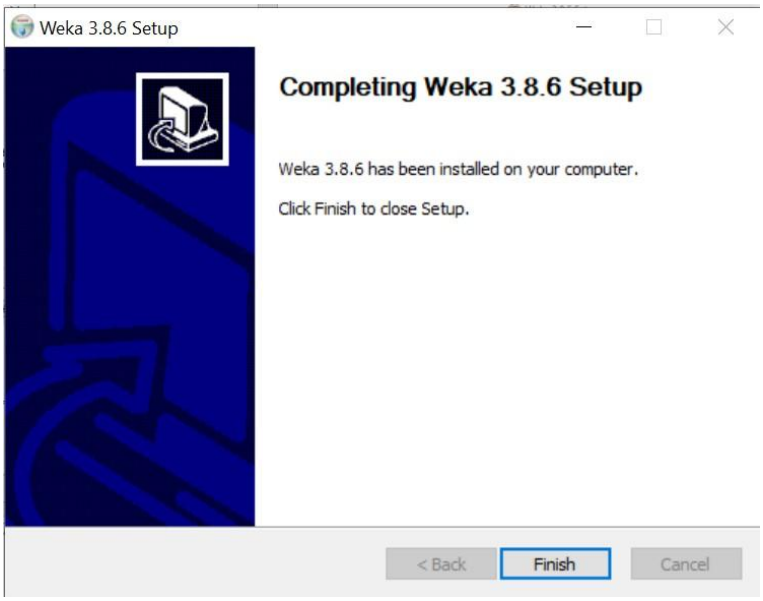
4. According to your requirements, select the components to be installed. Full component installation is recommended. Click on Next.



5. Select the destination folder and click on Next.
6. Click Install
7. Then, Installation will start.

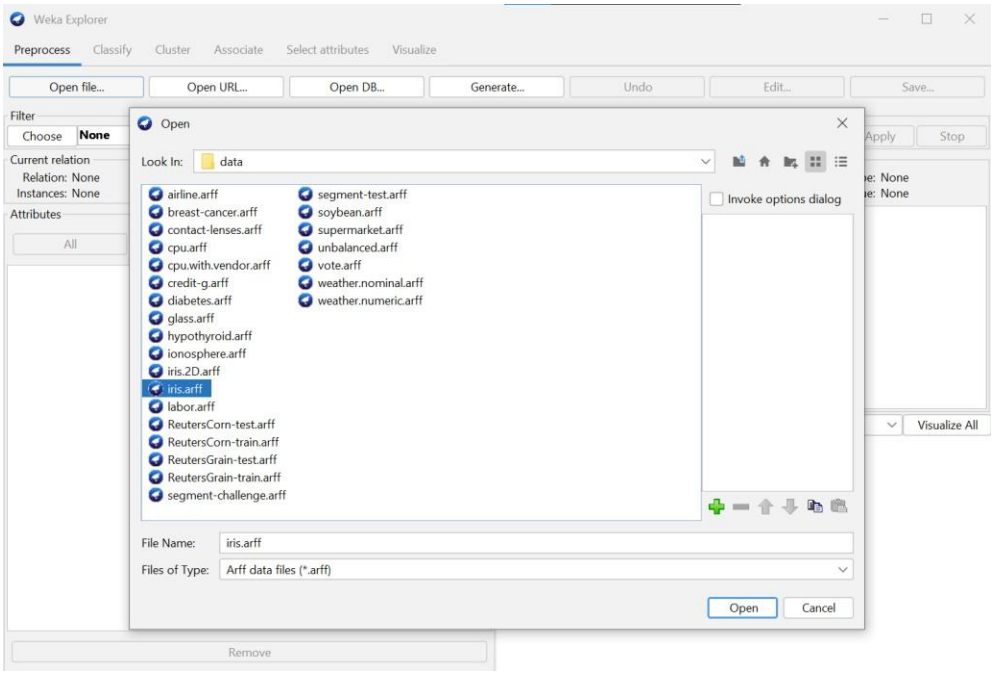


8. If Java is not installed in the system, it will install Java first.
9. After the installation is complete, the following window will appear. Click on Next.
10. Select the Start Weka checkbox. Click on Finish.



TASK 13: Data Visualization Using WEKA Explorer

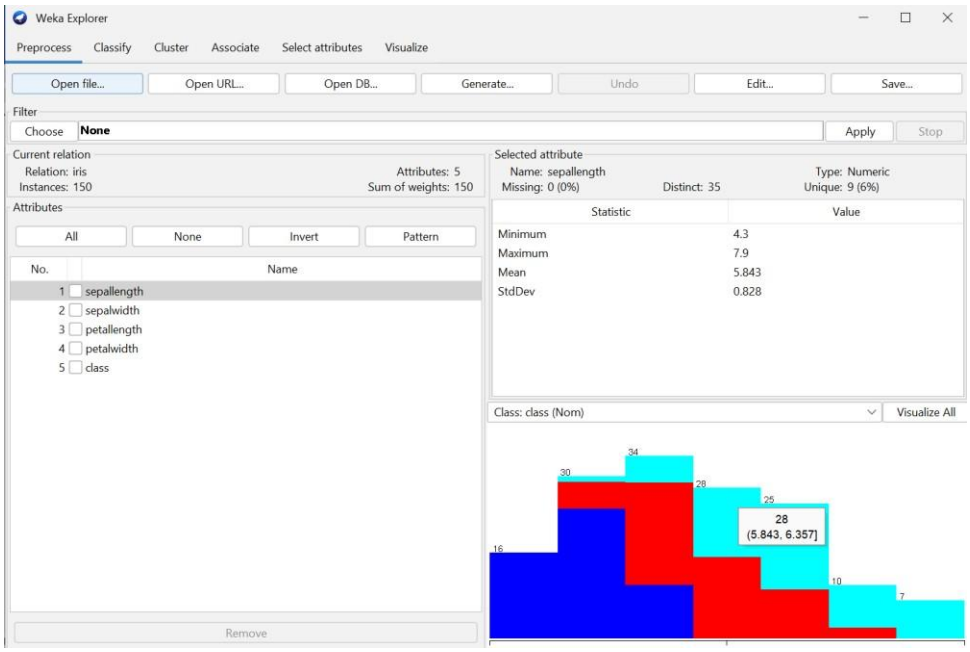
1. Go to the Preprocess tab and open **iris.arff** dataset. (Dataset path: C:\Program Files\ Weka-3-85\data\iris.arff)



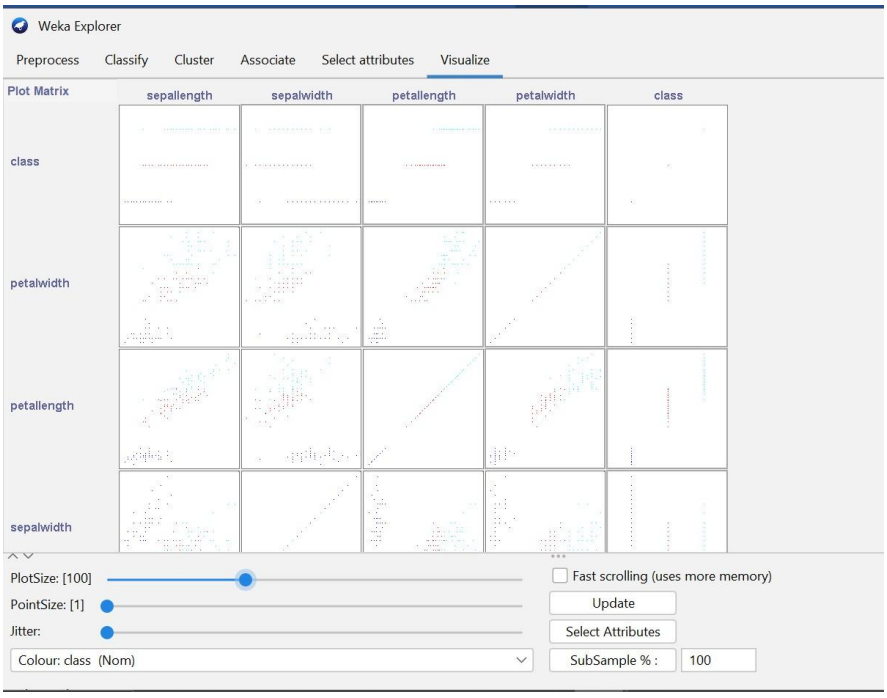
2. The dataset has 4 attributes and 1 class label. The attributes in this dataset are:

    - **Sepallength:** Type -numeric

- **Sepalwidth:** Type- numeric
- **Petalength:** Type-numeric
- **Petalwidth:** Type-numeric
- **Class:** Type-nominal



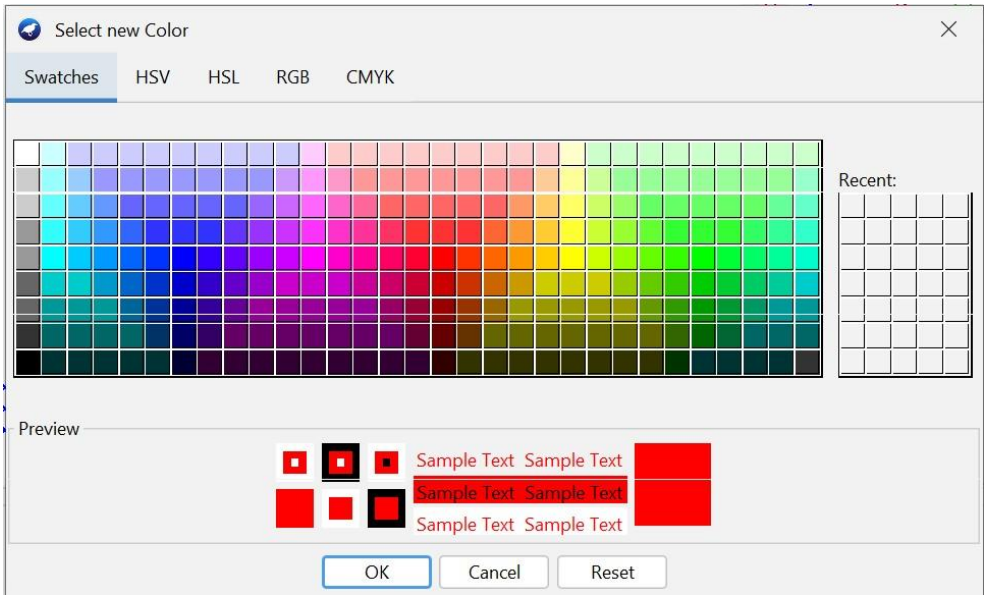
3. To visualize the dataset, go to the Visualize tab. The tab shows the attributes plot matrix. The dataset attributes are marked on the x-axis and y-axis while the instances are plotted. The box with x-axis attribute and y-axis attribute can be enlarged.



4. Click on the box of the plot to enlarge. **For example**, x: petallength and y: petalwidth. The class labels are represented in different colors.

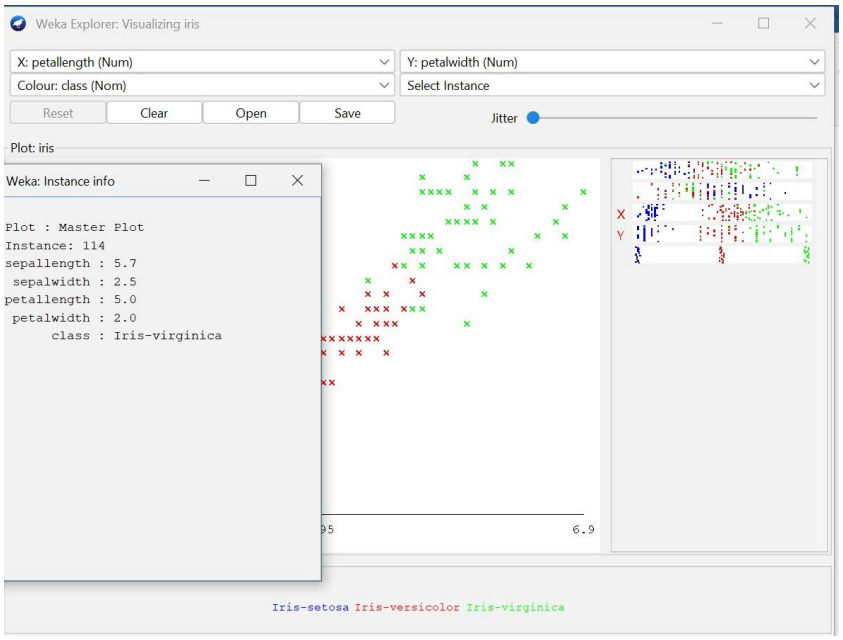
    - Class label- Iris-setosa: blue color
    - Class label- Iris-versicolor: red
    - Class label-Iris-virginica-green

These colors can be changed. To change the color, click on the class label at the bottom, a color window will appear.



5. Click on the instance represented by ‘x’ in the plot. It will give the instance details. For example:

- **Instance number**: 114
- **Sepalength:** 5.7
- **Sepalwidth:** 2.5
- **Petalength:** 5.0
- **Petalwidth:** 2.0
- **Class:** Iris-verginica



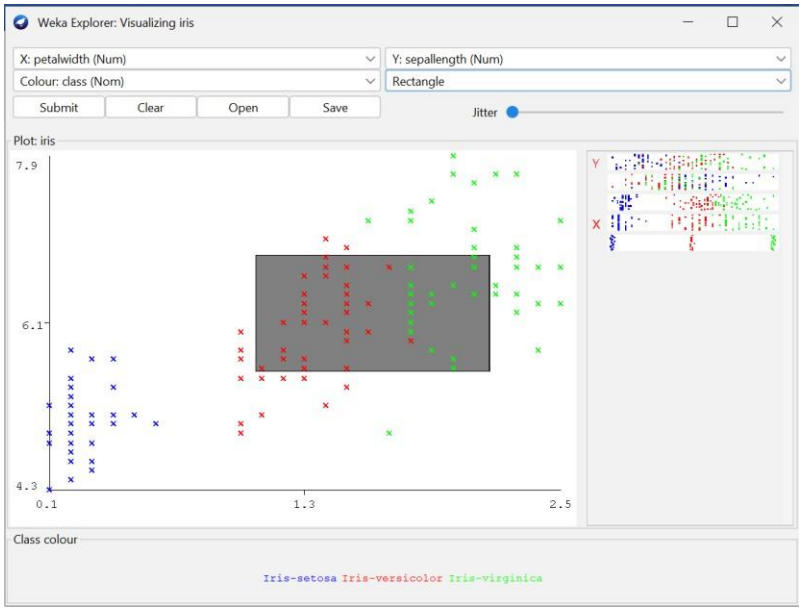
Some of the points in the plot appear darker than other points. These points represent 2 or more instances with the same class label and the same value of attributes plotted on the graph such as petalwidth and petallength.

The figure below represents a point with 2 instance information.

6. The X and Y-axis attributes can be changed from the right panel in Visualize graph. The user can view different plots.

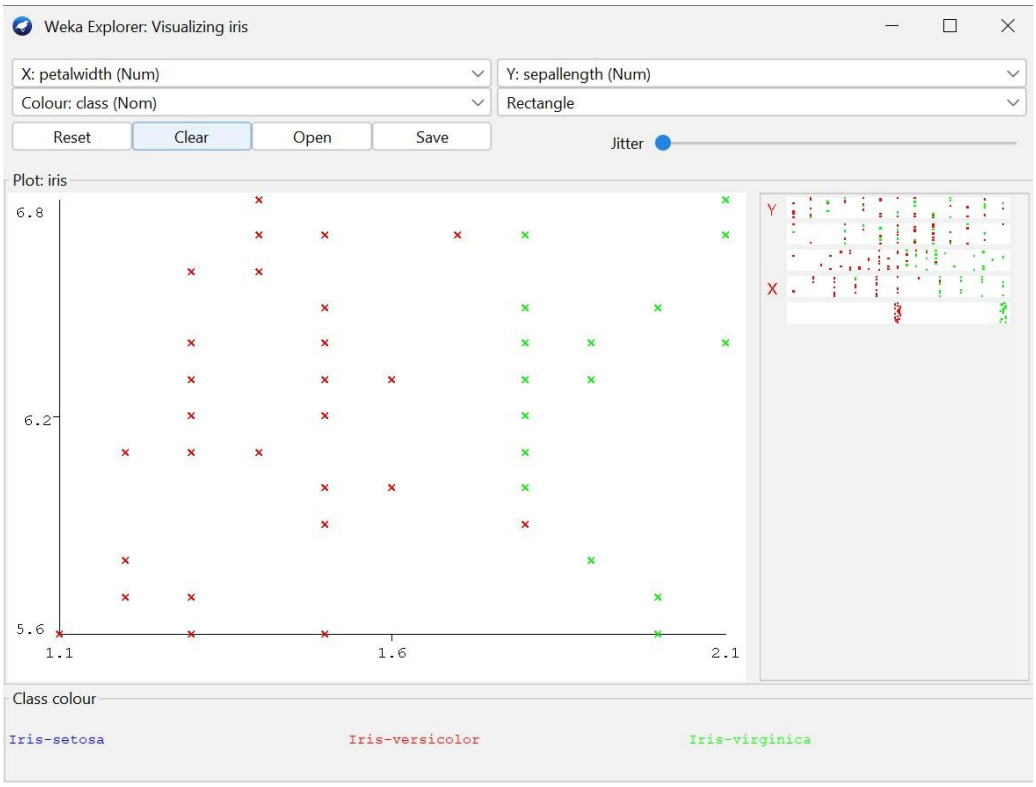
7. The Jitter is used to add randomness to the plot. Sometimes the points overlap. With jitter, the darker spots represent multiple instances.

8. To get a clearer view of the dataset and remove outliers, the user can select an instance from the dropdown. Click on “select instance” dropdown. Choose “Rectangle”. With this, the user will be able to select points in the plot by plotting a rectangle.



9. Click on “Submit”. Only the selected dataset points will be displayed and the other points will be excluded from the graph.

The figure below shows the points from the selected rectangular shape. The plot represents points with only 3 class labels. The user can click on “Save” to save the dataset or “Reset” to select another instance. The dataset will be saved in a separate .ARFF file.



Output:

Data visualization using WEKA is simplified with the help of the box plot. The user can view any level of granularity. The attributes are plotted on X-axis and y-axis while the instances are plotted against the X and Y-axis. Some points represent multiple instances which are represented by points with dark color.