PRACTICAL JOURNAL

in

DATA MANAGEMENT AND DATA WAREHOUSING

Submitted by

KSMSCIT005 HITESH VERSHI BHANUSHALI

for the award of the Degree of

MASTERS OF SCIENCE (INFORMATION TECHNOLOGY)

PART – II

DEPARTMENT OF INFORMATION TECHNOLOGY
KISHINCHAND CHELLARAM COLLEGE
(Affiliated to University of HSNCU)
MUMBAI, 400020
MAHARASHTRA
2024-25

SUBJECT CODE - BIT615D

DATA MANAGEMENT AND DATA WAREHOUSING

KISHINCHAND CHELLARAM COLLEGE

CHURCHGATE, MUMBAI – 400 020.

DEPARTMENT OF INFORMATION TECHNOLOGY

M.SC.I.T PART- II

CERTIFICATE

This is to certify that the Practical conducted by Mr. **HITESH VERSHI BHANUSHALI** for M.Sc. (IT) Part- II Semester- IV, Seat No: **KSMSCIT005** at Kishinchand Chellaram College in partial fulfillment for the MASTERS OF SCIENCE (INFORMATION TECHNOLOGY). Degree Examination for Semester II has been periodically examined and signed, and the course of term work has been satisfactorily carried out for the year 2024 - 2025. This Practical journal had not been submitted for any other examination and does not form part of any other course undergone by the candidate.

Signature	**Signature**	**Signature**
Lecturer-In-Charge	**External Examiner**	**Course Coordination**
Guided By	**Examined By**	**Certified By**

College Stamp

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester III

INDEX

Sr. No.	TITLE	Date	Signature
	PRACTICALS		
1	A) Data Pre-Processing and Exploration	01/12/2024	
	B) Load a Dataset, Calculate Descriptive Summary Statistics, Create Visualizations	08/12/2024	
	C) Data Pre-Processing Routines – Label Encoding, Scaling, and Binarization	08/12/2024	
2	A) Implementation of Decision Tree Learning	14/12/2024	
	B) Implementation of Decision Tree Learning Using Gini Index	21/12/2024	
	C) Implementation of Gini Index in Python	21/12/2024	
	D) Implementation of Decision Tree Using Information Gain	21/12/2024	
	E) Implementation of Information Gain	21/12/2024	
3	A) Implementation of Naïve Bayes Classifier (Spam vs. Ham Emails)	28/12/2024	
	B) Implementation of Naïve Bayes Classifier (Iris Dataset)	28/12/2024	
4	A) Reading Data from Different Files Using R	04/01/2025	
	B) Implementing Classification in R (Decision Tree)	05/01/2025	
	C) Implementing Classification in R (Naïve Bayes)	05/01/2025	
5	A) Implementing Classifier in Python (SVM)	16/02/2025	
	B) Implementing Classifier in R (SVM)	16/02/2025	
6	A) Implementation of K-Means in Python	23/02/2025	
	B) Implementation of K-Means in R	23/02/2025	
7	Understanding Weka	16/03/2025	
8	Implementing Classification in Weka (Decision Tree)	16/03/2025	
9	Implementing Classification in Weka (Naïve Bayes)	16/03/2025	
10	Implementing Clustering with Weka	16/03/2025	
11	Assignment		

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester IV

Practical 1A: Data Preprocessing and Exploration

Link to download Dataset
https://drive.google.com/file/d/1XWibmqRp2lGuoiqUW523SgMB6z4Wr2IW/view?usp=sh aring

Display Data:

```
import pandas as pd
df = pd.read_csv("/content/SampleData.csv")
print(df)
```

```
   Sr No     Name  Age          City     Occupation
0      1    Alice   25      New York       Engineer
1      2      Bob   30   Los Angeles       Designer
2      3  Charlie   28       Chicago        Teacher
3      4    David  200       Houston         Doctor
4      5    Glory   22       Phoenix   Data Analyst
5      6    Frank   40  Philadelphia         Lawyer
6      7    Grace   29   San Antonio      Scientist
7      8   Hannah   31     San Diego      Architect
8      9    Isaac   27        Dallas      Developer
9     10    Julia   33      San Jose      Marketing
```

→ **Handle missing values :** To handle missing values first we delete two values from names Alice and Emma from SampleData.csv and save it and again upload on google colab, Now the data set is

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Sr No       10 non-null     int64
 1   Name        10 non-null     object
 2   Age         10 non-null     int64
 3   City        10 non-null     object
 4   Occupation  10 non-null     object
dtypes: int64(2), object(3)
memory usage: 528.0+ bytes
None
```

```
print(df["Name"])
```

```
0     Ashish
1        Bob
2    Charlie
3      David
4     Ashish
5      Frank
6      Grace
7     Hannah
8      Isaac
9      Julia
Name: Name, dtype: object
```

```
print(df["Name"].isnull())

0    False
1    False
2    False
3    False
4    False
5    False
6    False
7    False
8    False
9    False
Name: Name, dtype: bool
```

→ **Handling inconsistent formatting:** Handling inconsistent formatting in datasets is a crucial part of data preprocessing. Inconsistent Formatting can manifest in various ways, such as differing text case, leading/trailing whitespaces, or inconsistent date formats. Below are common techniques in Python to address these issues using the pandas library.

1. **Standardizing Text Data**
a. Convert to Lowercase or Uppercase

```
df["Name"]=df["Name"].str.lower()
print(df["Name"])

0     ashish
1        bob
2    charlie
3      david
4     ashish
5      frank
6      grace
7     hannah
8      isaac
9      julia
Name: Name, dtype: object
```

```
df["Name"]=df["Name"].str.upper()
print(df["Name"])

0     ASHISH
1        BOB
2    CHARLIE
3      DAVID
4     ASHISH
5      FRANK
6      GRACE
7     HANNAH
8      ISAAC
9      JULIA
Name: Name, dtype: object
```

b. **Remove Leading/Trailing Whitespaces (Fixes issues with extra spaces.)**

```
df["Name"]=df["Name"].str.strip()
print(df["Name"])

0     ASHISH
1        BOB
2    CHARLIE
3      DAVID
4     ASHISH
5      FRANK
6      GRACE
7     HANNAH
8      ISAAC
9      JULIA
Name: Name, dtype: object
```

→ **Handling outliers**

What Are Outliers?

• Outliers are data points that deviate significantly from the rest of the dataset. They appear to be unusually large or small compared to other values and can result from variability in the data or errors in data collection, entry, or processing.

Characteristics of Outliers:

• Extreme Values: Outliers lie far from the typical range of values.
• Rare Occurrence: They are relatively few compared to the total number of data points.
• Potential Impact: Outliers can skew statistical metrics such as the mean, standard deviation, and even the results of machine learning models.

```python
# Practical 1A: Data Preprocessing and Exploration

import pandas as pd
import numpy as np
# Sample data
data = {'Value': [10, 12, 15, 14, 13, 12, 10, 11, 12, 14, 10]}
df = pd.DataFrame(data)
#Calculate IQR
Q1=df['Value'].quantile (0.25)
Q3=df['Value'].quantile (0.75)
IQR = Q3 - Q1
#Define bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
#Identify outliers
outliers =df[(df[ 'Value'] < lower_bound) | (df[ 'Value'] > upper_bound)]
print("Hitesh Bhanushali KSMSCIT005")
print("---------------------")
print("Outliers: \n", outliers)
```

```
Hitesh Bhanushali KSMSCIT005
---------------------
Outliers:
 Empty DataFrame
Columns: [Value]
Index: []
```

```python
# Practical 1A: Data Preprocessing and Exploration
import pandas as pd
df = pd.read_csv("/content/SampleData.csv")
data = df["Age"]
dataInt = data.astype(int)
datatolist= dataInt.tolist()
print(datatolist)
#Calculate IQR
Q1= dataInt.quantile (0.25)
Q3= dataInt.quantile (0.75)
IQR=Q3 -Q1
# Define bounds
lower_bound = Q1 - 1.5 *IQR
upper_bound =Q3 + 1.5 * IQR
#Identify outliers
outliers =dataInt [(dataInt<lower_bound) | (dataInt > upper_bound)]
print("Hitesh Bhanushali KSMSCIT005")
print("-------")
print(outliers)
```

```
[25, 30, 28, 200, 22, 40, 29, 31, 27, 33]
Hitesh Bhanushali KSMSCIT005
-------
3    200
Name: Age, dtype: int64
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester IV

Practical 1B: Load a Dataset, Calculate Descriptive Summary Statistics, Create Visualizations.

Note: For this practical we use train.csv dataset

Link to download train.csv dataset is

https://drive.google.com/file/d/1owBsqmE23cF9rPa0AYeBradABtKIwAS6/view?usp=sharing

Load a dataset:

```python
# Practical 1B: Load a Dataset, Calculate Descriptive Summary Statistics, Create Visualizations.

import pandas as pd
print("Hitesh Bhanushali KSMSCIT005")
print("-------")
df = pd.read_csv("/content/train.csv")
df.head()
```

```
Hitesh Bhanushali KSMSCIT005
-------
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	ST
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	

Calculate Descriptive Summary Statistics

```python
print("Hitesh Bhanushali KSMSCIT005")
print("-------")
print(df.describe())
```

```
Hitesh Bhanushali KSMSCIT005
-------
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
```

Additional Summary Statistics:

For all columns, including non-numeric:

```
[ ] print('Hitesh Bhanushali KSMSCIT005')
    print("==============================")
    print(df.describe())

    print(df.describe(include= 'all'))
```

```
Hitesh Bhanushali KSMSCIT005
==============================
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
```

Identify potential features and target variables

```
[ ] print('Hitesh Bhanushali KSMSCIT005')
    print("=============================")
    x = df.drop(['Survived'], axis=1)
    y = df['Survived']

    print("Features: \n",x.head())
    print("Target: ",y.head())
```

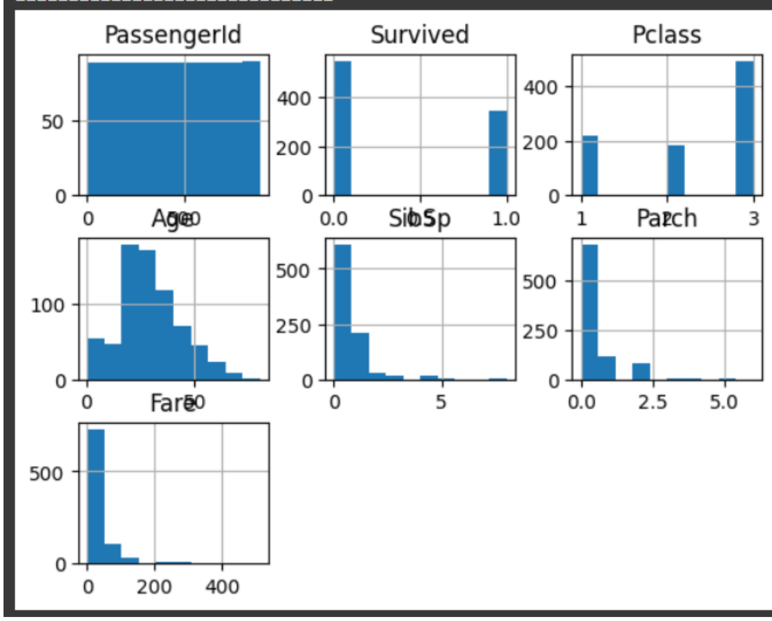
```
Hitesh Bhanushali KSMSCIT005
=============================
Features:
    PassengerId  Pclass                                               Name  \
0            1       3                            Braund, Mr. Owen Harris
1            2       1  Cumings, Mrs. John Bradley (Florence Briggs Th...
2            3       3                             Heikkinen, Miss. Laina
3            4       1       Futrelle, Mrs. Jacques Heath (Lily May Peel)
4            5       3                           Allen, Mr. William Henry

      Sex   Age  SibSp  Parch            Ticket     Fare Cabin Embarked
0    male  22.0      1      0         A/5 21171   7.2500   NaN        S
1  female  38.0      1      0          PC 17599  71.2833   C85        C
2  female  26.0      0      0  STON/O2. 3101282   7.9250   NaN        S
3  female  35.0      1      0            113803  53.1000  C123        S
4    male  35.0      0      0            373450   8.0500   NaN        S
Target:  0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

Create visualizations using different graphs Histogram

The **histogram** represents the frequency of occurrence of specific phenomena which lie within a specific range of values and are arranged in consecutive and fixed intervals.

```
] # Create visualizations using different graphs
  # Histogram

  import matplotlib.pyplot as plt

  #create histogram for numeric data
  df.hist()
  print('Hitesh Bhanushali KSMSCIT005')
  print("==============================")
  plt.show()
```

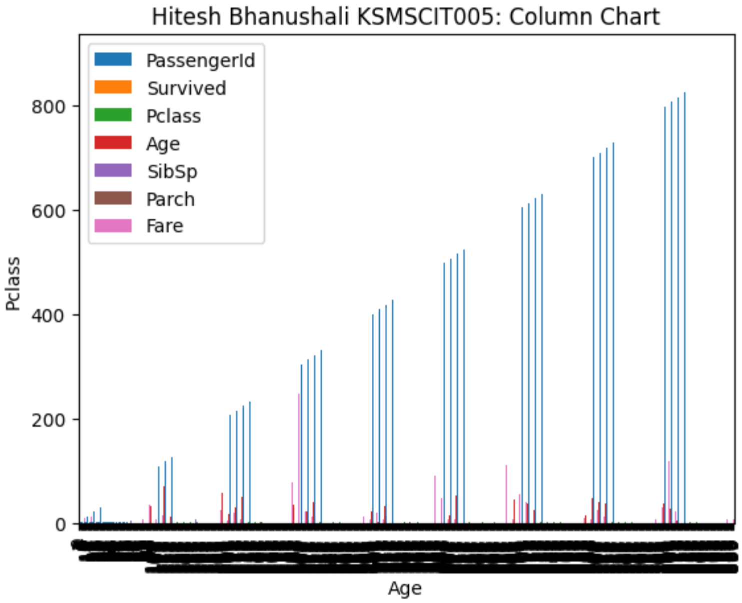
```
Hitesh Bhanushali KSMSCIT005
==============================
```



Column Chart

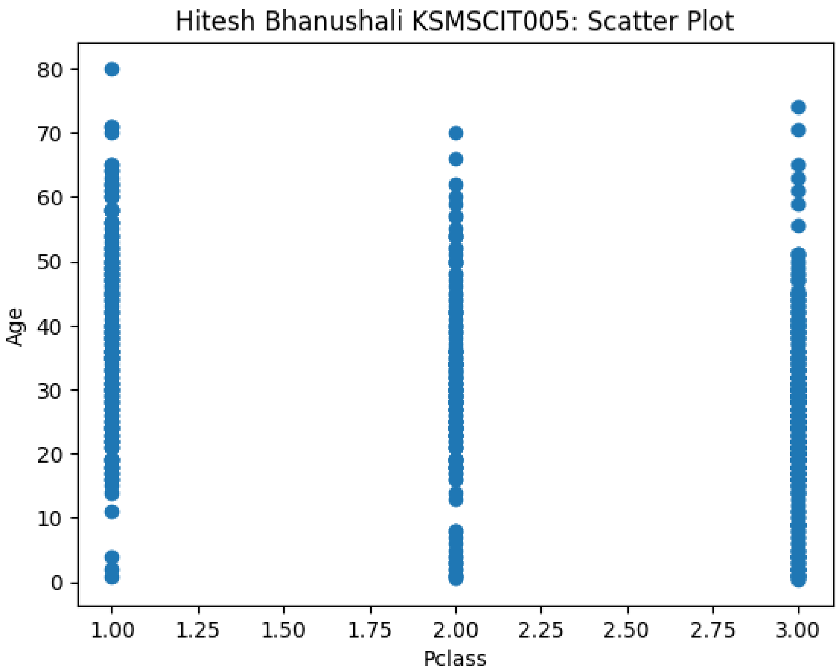
A column chart is used to show a comparison among different attributes, or it can show a comparison of items over time.

```
# Column Chart

print('Hitesh Bhanushali KSMSCIT005')
print("==============================")
df.plot.bar()
plt.bar(df['Age'], df['Pclass'])
plt.xlabel('Age')
plt.ylabel('Pclass')
plt.title('Hitesh Bhanushali KSMSCIT005: Column Chart')
plt.show()
```



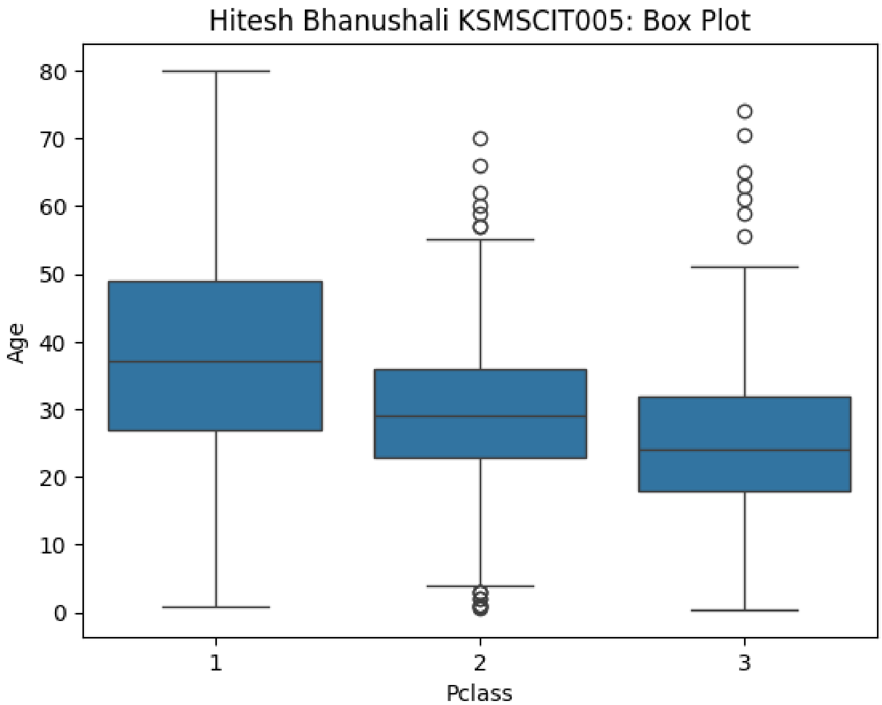
Scatter Plot

```
#scatter plot
print('Hitesh Bhanushali KSMSCIT005')
print("==============================")
plt.scatter(df['Pclass'], df['Age'])
plt.xlabel('Pclass')
plt.ylabel('Age')
plt.title('Hitesh Bhanushali KSMSCIT005: Scatter Plot')
plt.show()
```



Box Plot

```
import seaborn as sns
sns.boxplot(x='Pclass', y='Age', data=df)
plt.title('Hitesh Bhanushali KSMSCIT005: Box Plot')
plt.show()
```



Practical 1C: Data Pre-Processing Routines – Label Encoding, Scaling, and Binarization

Note: For this practical we use iris.csv dataset

Link to download train.csv dataset is
https://drive.google.com/drive/folders/1aLLSp0Y5ZlL2iVcUt9TfmNWM7i4QEOQ?usp=sharing

Example of Label Encoding
We will apply Label Encoding on the iris dataset on the target column which is Species. It contains three species Iris-setosa, Iris-versicolor, Iris-virginica.

```
import numpy as np
import pandas as pd

df = pd.read_csv("/content/iris (3).csv")
print('Hitesh Bhanushali KSMSCIT005')
print("==============================")
print(df.head())
print(df['species'].unique())
```

```
Hitesh Bhanushali KSMSCIT005
==============================
   sepal_length  sepal_width  petal_length  petal_width 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
['setosa' 'versicolor' 'virginica']
```

After applying Label Encoding with LabelEncoder() our categorical value will replace with the numerical value[int].

```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
df['species']= label_encoder.fit_transform(df['species'])
print('Hitesh Bhanushali KSMSCIT005')
print("==============================")
print(df['species'].unique())
```

```
Hitesh Bhanushali KSMSCIT005
==============================
[0 1 2]
```

```
#Binarization

from sklearn.preprocessing import Binarizer
from sklearn.datasets import load_iris
import pandas as pd
import numpy as np

data = load_iris(as_frame=True)
print('Hitesh Bhanushali KSMSCIT005')
print("==============================")
df = data.frame
# df.head()
print(df['sepal length (cm)'])
```

```
Hitesh Bhanushali KSMSCIT005
==============================
0      5.1
1      4.9
2      4.7
3      4.6
4      5.0
      ...
145    6.7
146    6.3
147    6.5
148    6.2
149    5.9
Name: sepal length (cm), Length: 150, dtype: float64
```

After Applying Binarization

```
##After Applying Binarization
binarizer = Binarizer(threshold=5)
df['sepal length (cm) binary'] = binarizer.fit_transform(df[['sepal length (cm)']])
print('Hitesh Bhanushali KSMSCIT005')
print("==============================")
print(df['sepal length (cm) binary'])
```

```
Hitesh Bhanushali KSMSCIT005
==============================
0      1.0
1      0.0
2      0.0
3      0.0
4      0.0
      ...
145    1.0
146    1.0
147    1.0
148    1.0
149    1.0
Name: sepal length (cm) binary, Length: 150, dtype: float64
```

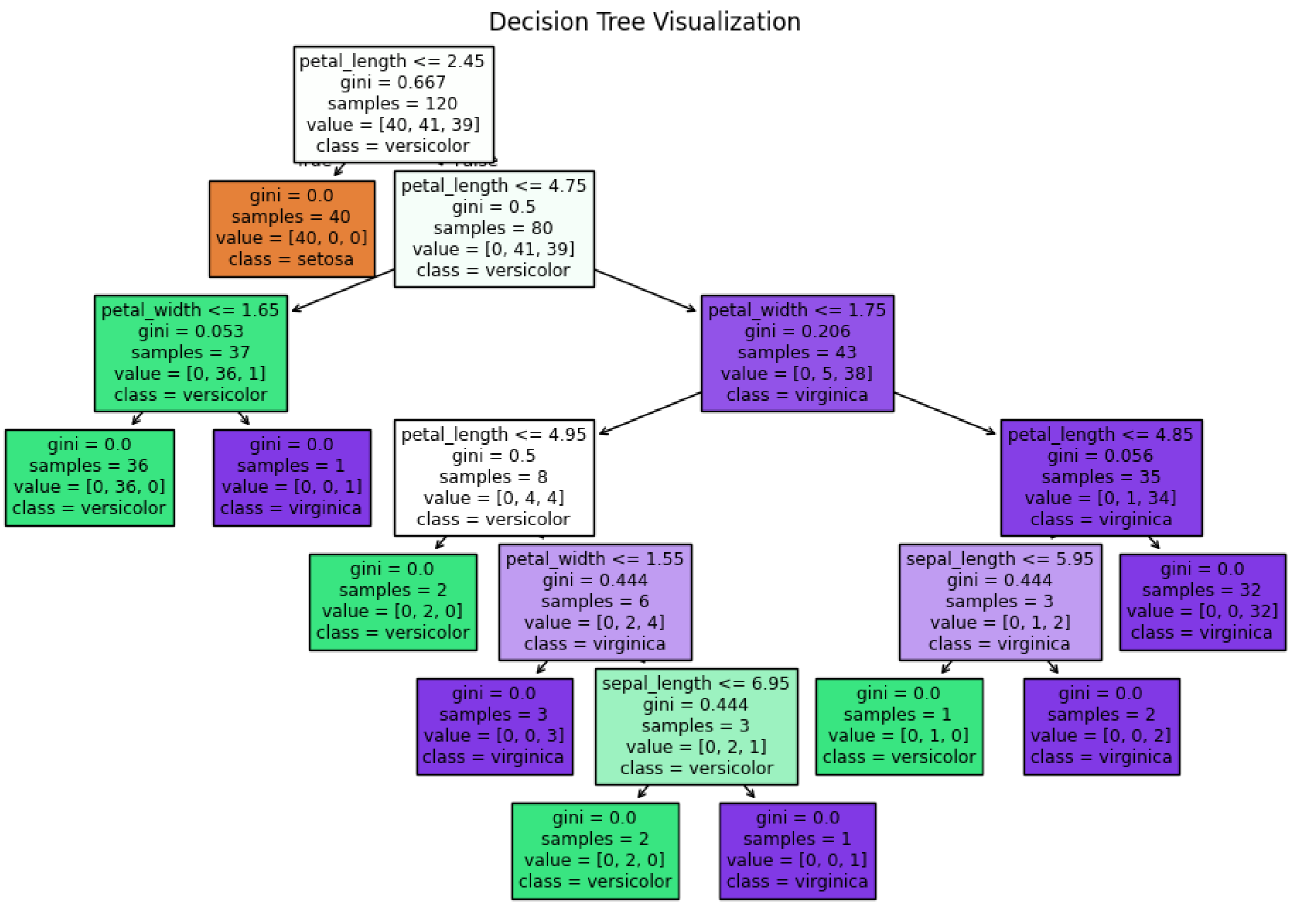
Practical 2A: Implementation of Decision Tree using Information Gain

Code:

```
# Practical 2
# Import necessary libraries
import numpy as np
import pandas as pd  # Import Pandas for data loading
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
print("Hitesh Bhanushali KSMSCIT005")
print("========================")
# Step 2: Load the dataset
data = pd.read_csv('/content/iris.csv')
# Step 3: Define features (X) and target variable (y)
X = data.drop('species', axis=1)  # Features (all columns except 'species')
y = data['species']  # Target variable ('species' column)
# Step 4: Split the dataset into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Step 5: Create and train a Decision Tree classifier
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)  # Train the model on training data
# Step 6: Make predictions on the test data
y_pred = clf.predict(X_test)
# Step 7: Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
# Step 8: Visualize and interpret the generated decision tree
plt.figure(figsize=(12, 8))
plot_tree(clf, filled=True, feature_names=X.columns, class_names=y.unique().astype(str))
plt.title("Decision Tree Visualization")
plt.show()
```

Output:

```
Hitesh Bhanushali KSMSCIT005
========================
Accuracy: 1.00
```



Decision Tree Visualization
petal_length <= 2.45
gini = 0.667
samples = 120
value = [40, 41, 39]
class = versicolor
gini = 0.0
samples = 40
value = [40, 0, 0]
class = setosa
petal_length <= 4.75
gini = 0.5
samples = 80
value = [0, 41, 39]
class = versicolor
petal_width <= 1.65
gini = 0.053
samples = 37
value = [0, 36, 1]
class = versicolor
petal_width <= 1.75
gini = 0.206
samples = 43
value = [0, 5, 38]
class = virginica
gini = 0.0
samples = 36
value = [0, 36, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
petal_length <= 4.95
gini = 0.5
samples = 8
value = [0, 4, 4]
class = versicolor
petal_length <= 4.85
gini = 0.056
samples = 35
value = [0, 1, 34]
class = virginica
gini = 0.0
samples = 2
value = [0, 2, 0]
class = versicolor
petal_width <= 1.55
gini = 0.444
samples = 6
value = [0, 2, 4]
class = virginica
sepal_length <= 5.95
gini = 0.444
samples = 3
value = [0, 1, 2]
class = virginica
gini = 0.0
samples = 32
value = [0, 0, 32]
class = virginica
gini = 0.0
samples = 3
value = [0, 0, 3]
class = virginica
sepal_length <= 6.95
gini = 0.444
samples = 3
value = [0, 2, 1]
class = versicolor
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
gini = 0.0
samples = 2
value = [0, 2, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester IV

Practical 2B: Implementation of Decision Tree Learning using Gini Index

Code:

```
# Step 1: Import necessary libraries
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, export_text, plot_tree
import matplotlib.pyplot as plt
data = load_iris()
X, y = data.data, data.target
# Step 2: Train Decision Tree with Gini Index
clf = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=42).fit(X, y)
# Step 3: Visualize Decision Tree
plt.figure(figsize=(10, 6))
plot_tree(clf, feature_names=data.feature_names, class_names=data.target_names,
filled=True, rounded=True)
plt.title("Hitesh Bhanushali: Decision Tree (Using Gini Index)")
plt.show()
# Step 4: Print Decision Tree rules
print(export_text(clf, feature_names=data.feature_names))
print("Hitesh Bhanushali KSMSCIT005")
print("========================")
```

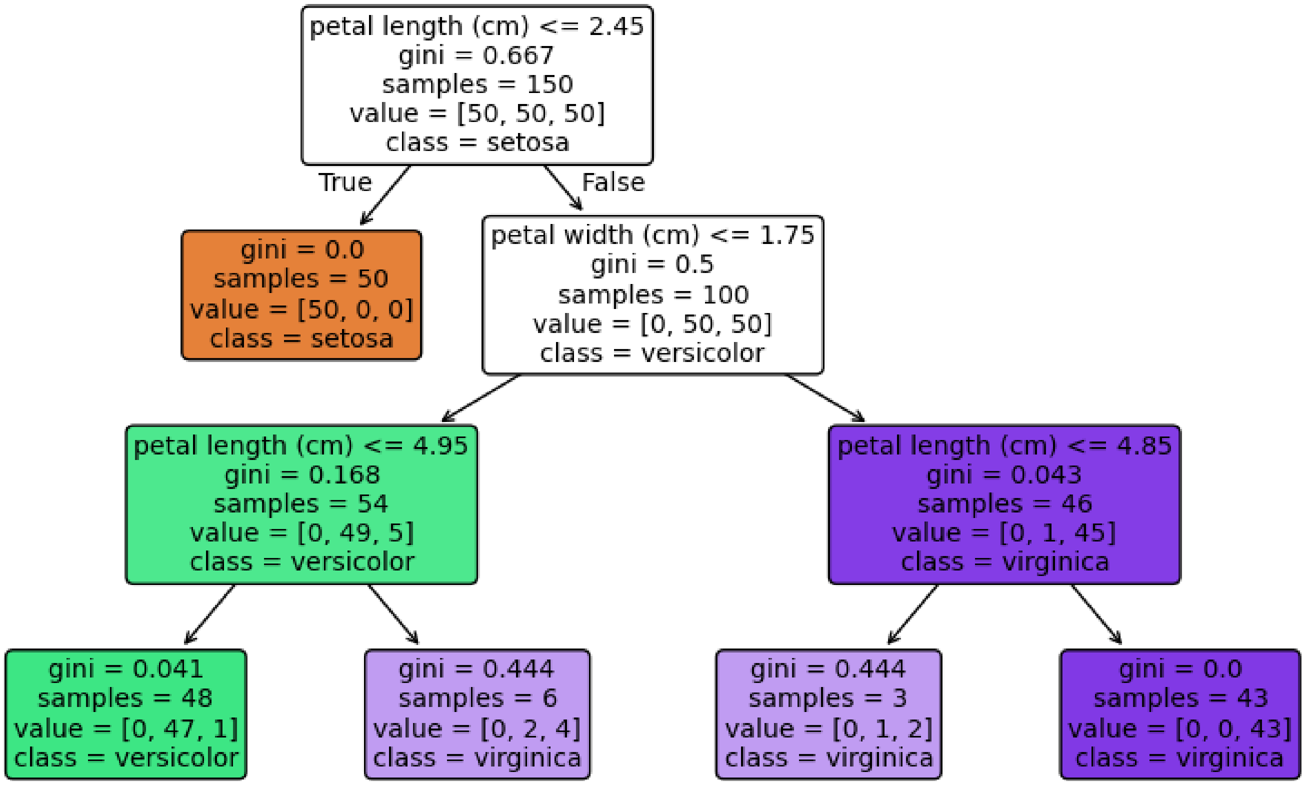
OUTPUT:

```
Hitesh Bhanushali KSMSCIT005
========================
|--- petal length (cm) <= 2.45
|   |--- class: 0
|--- petal length (cm) >  2.45
|   |--- petal width (cm) <= 1.75
|   |   |--- petal length (cm) <= 4.95
|   |   |   |--- class: 1
|   |   |--- petal length (cm) >  4.95
|   |   |   |--- class: 2
|   |--- petal width (cm) >  1.75
|   |   |--- petal length (cm) <= 4.85
|   |   |   |--- class: 2
|   |   |--- petal length (cm) >  4.85
|   |   |   |--- class: 2
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester IV

Hitesh Bhanushali: Decision Tree (Using Gini Index)



Practical 2C: Implementing of Gini Index in Python

Code:

```
# Step 1: Gini Index Calculation Functions
def gini_index(classes):
    total = sum(classes)
    return 1 - sum((count / total) ** 2 for count in classes)
print("Hitesh Bhanushali KSMSCIT005\n=========================")
def weighted_gini(children):
    total_instances = sum(sum(child) for child in children)
    return sum((sum(child) / total_instances) * gini_index(child) for child in children)
def gini_gain(parent, children):
    return gini_index(parent) - weighted_gini(children)
# Step 2: Example: Dataset
parent, child_1, child_2 = [18, 33, 10], [16, 26, 24], [14, 25, 11]
# Step 3: Calculations & Results
print(f"Gini Index (Parent): {gini_index(parent):.4f}")
print(f"Weighted Gini Index (Split): {weighted_gini([child_1, child_2]):.4f}")
print(f"Gini Gain: {gini_gain(parent, [child_1, child_2]):.4f}")
print("Hitesh Bhanushali KSMSCIT005\n=========================")
```

Output:

```
Hitesh Bhanushali KSMSCIT005
=========================
Gini Index (Parent): 0.5934
Weighted Gini Index (Split): 0.6406
Gini Gain: -0.0472
Hitesh Bhanushali KSMSCIT005
=========================
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester IV

Practical 2D: Implementation of Decision Tree using Information gain

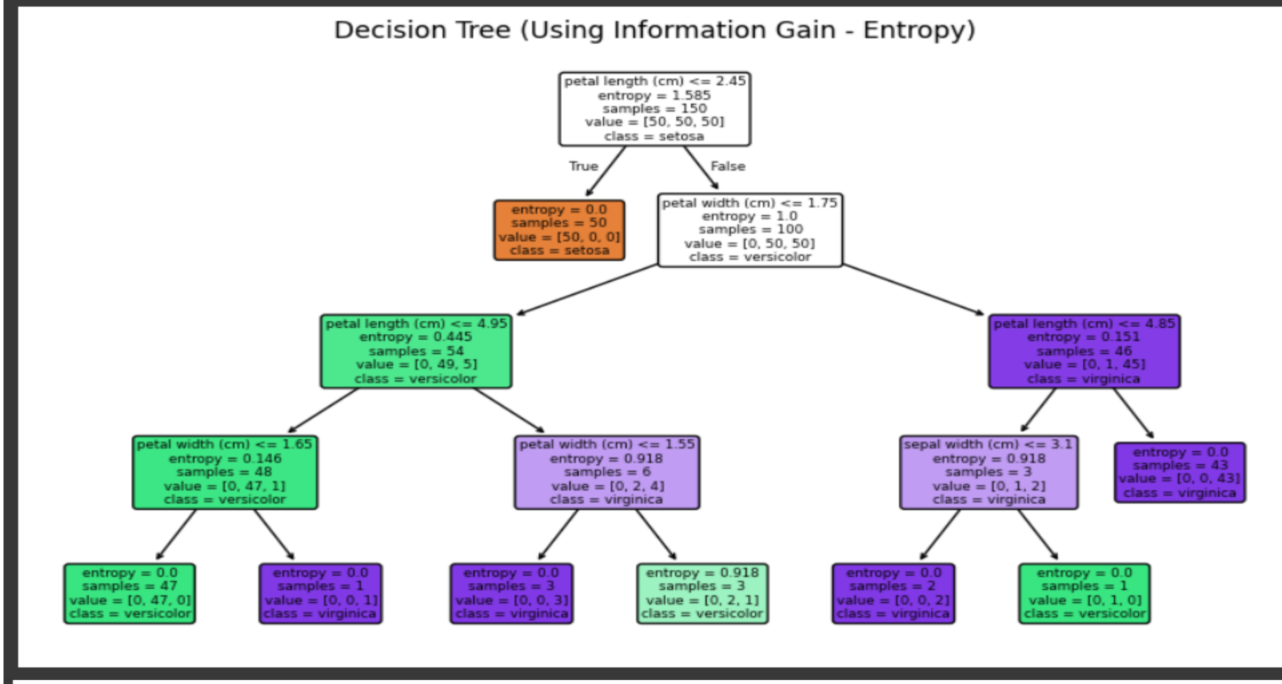
Step 1: Import Dataset and required libraiers

Code:

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, export_text, plot_tree
import matplotlib.pyplot as plt
data = load_iris()
X, y = data.data, data.target
#Step 2: Train Decision Tree with Information Gain (Entropy)
clf = DecisionTreeClassifier(criterion='entropy', max_depth=4, random_state=42).fit(X,
y)
#Step 3: Visualize Decision Tree
plt.figure(figsize=(10, 6))
print("Hitesh Bhanushali KSMSCIT005\n=========================")
plot_tree(clf, feature_names=data.feature_names, class_names=data.target_names,
filled=True, rounded=True)
plt.title("Decision Tree (Using Information Gain - Entropy)")
plt.show()

#Step 4: Print the decision tree in text
tree_rules = export_text(clf, feature_names=data.feature_names)
print("Hitesh Bhanushali KSMSCIT005\n=========================")
print(tree_rules)
```

Output:

```
Hitesh Bhanushali KSMSCIT005
=========================
```




```
Hitesh Bhanushali KSMSCIT005
=========================
|--- petal length (cm) <= 2.45
|   |--- class: 0
|--- petal length (cm) >  2.45
|   |--- petal width (cm) <= 1.75
|   |   |--- petal length (cm) <= 4.95
|   |   |   |--- petal width (cm) <= 1.65
|   |   |   |   |--- class: 1
|   |   |   |--- petal width (cm) >  1.65
|   |   |   |   |--- class: 2
|   |   |--- petal length (cm) >  4.95
|   |   |   |--- petal width (cm) <= 1.55
|   |   |   |   |--- class: 2
|   |   |   |--- petal width (cm) >  1.55
|   |   |   |   |--- class: 1
|   |--- petal width (cm) >  1.75
|   |   |--- petal length (cm) <= 4.85
|   |   |   |--- sepal width (cm) <= 3.10
|   |   |   |   |--- class: 2
|   |   |   |--- sepal width (cm) >  3.10
|   |   |   |   |--- class: 1
|   |   |--- petal length (cm) >  4.85
|   |   |   |--- class: 2
```

<u>Practical 2E: Implementation of Information gain</u>

Code:

```
import numpy as np
def entropy (classes):
  total = sum(classes)
  proportions = [count / total for count in classes if count > 0]
  return -sum(p* np.log2 (p) for p in proportions)
def information_gain (parent, children):
  total_instances = sum(parent)
  parent_entropy = entropy(parent)
  weighted_entropy = sum(
    (sum(child) / total_instances) * entropy (child) for child in children
  )
  return parent_entropy, weighted_entropy
# Dataset
# Class A: 50,
#Class B: 30, Class C: 20 # Class A: 30, Class B: 20, Class C: 10 # Class A: 20, Class B: 10, Class C: 10

parent_node = [50, 30, 20]
child_1 = [30, 20, 10]
child_2 = [20, 10, 10]

# Calculations
parent_entropy = entropy (parent_node)
weighted_entropy = sum([entropy (child_1), entropy (child_2)])
gain= information_gain (parent_node, [child_1, child_2])
# Results
print("Hitesh Bhanushali KSMSCIT005\n=========================")
print (f"Entropy (Parent Node): {parent_entropy:.4f}")
print (f"Weighted Entropy (After Split): {weighted_entropy:.4f}")
print (f"Information Gain: (gain:.4f)")
print("=======")
```

Output:

```
Hitesh Bhanushali KSMSCIT005
=========================
Entropy (Parent Node): 1.4855
Weighted Entropy (After Split): 2.9591
Information Gain: (gain:.4f)
=======
```

Practical 3A: Classification of email as Spam or Ham

We will have a dataset of 6 emails with the following words: "free", "money", "win", "meeting", "hello". The labels will be 1 for spam and 0 for ham.

Email	free	money	win	meeting	hello	Label
Email 1	1	1	1	0	0	1 (Spam)
Email 2	1	1	0	0	0	1 (Spam)
Email 3	1	0	0	0	0	1 (Spam)
Email 4	0	0	0	1	1	0 (Ham)
Email 5	0	0	0	1	1	0 (Ham)
Email 6	0	0	0	1	1	0 (Ham)

Step 1: Implementation of Naïve Bayes Classifier for Spam Detection

Code:

```
import numpy as np
from sklearn.naive_bayes import BernoulliNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
print("Hitesh Bhanushali KSMSCIT005\n=========================")
```

Step 2: Prepare dataset (Binary representation of words: free, money, win, meeting)

```
X = np.array([
    [1, 1, 1, 0, 0],  # Spam
    [1, 1, 0, 0, 0],  # Spam
    [1, 0, 0, 0, 0],  # Spam
    [0, 0, 0, 1, 1],  # Ham
    [0, 0, 0, 1, 1],  # Ham
    [0, 0, 0, 1, 1],  # Ham
])
y = np.array([1, 1, 1, 0, 0, 0])  # Labels (1: Spam, 0: Ham)
```

```
Hitesh Bhanushali KSMSCIT005
=========================
Accuracy: 100.00%
Predicted class (0=Ham, 1=Spam): 0
Hitesh Bhanushali KSMSCIT005
=========================
```

Step 3: Train model and predict

```
#Step 02: Split the dataset 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)
model = BernoulliNB()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
# Step 3: Make predictions & evaluate accuracy
```

Step 4: Make predictions & evaluate accuracy

```
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
new_email = np.array([[0, 0, 0, 1, 1]])  # Example email # Step 4: Predict new email classification
print(f'Predicted class (0=Ham, 1=Spam): {model.predict(new_email)[0]}')
print("Hitesh Bhanushali KSMSCIT005\n=========================")
```

Practical 3B: Implementation of Naïve Bayes Classifier [Iris Dataset]

Code:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
X, y = load_iris(return_X_y=True)
print("Hitesh Bhanushali KSMSCIT005\n=========================")
# Split dataset & train classifier
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = GaussianNB().fit(X_train, y_train)
y_pred = model.predict(X_test) # Predictions & accuracy
print("Predicted Output:\n", y_pred)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Hitesh Bhanushali KSMSCIT005\n=========================")
```

```
Hitesh Bhanushali KSMSCIT005
=========================
Predicted Output:
 [1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
Accuracy: 1.0
Hitesh Bhanushali KSMSCIT005
=========================
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester IV

Practical 4A: Reading Data from Different Files Using [R Studio]

Code:

```
library(datasets)
data(iris)
print("Hitesh Bhanushali KSMSCIT005\n=========================")
print(iris)
names(iris)
summary(iris)
summary(iris$Sepal.Width)
is.na(iris$Sepal.Width)
is.na(iris)
length(unique(iris$Sepal.Width)) #To find the unique values
print("Hitesh Bhanushali KSMSCIT005\n=========================")
plot(iris$Sepal.Width)
```

```
[1] "Hitesh Bhanushali KSMSCIT005\n========================"
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
4            4.6         3.1          1.5         0.2     setosa
5            5.0         3.6          1.4         0.2     setosa
6            5.4         3.9          1.7         0.4     setosa
7            4.6         3.4          1.4         0.3     setosa
8            5.0         3.4          1.5         0.2     setosa
9            4.4         2.9          1.4         0.2     setosa
10           4.9         3.1          1.5         0.1     setosa
11           5.4         3.7          1.5         0.2     setosa
12           4.8         3.4          1.6         0.2     setosa
13           4.8         3.0          1.4         0.1     setosa
```

```
'Sepal.Length' · 'Sepal.Width' · 'Petal.Length' · 'Petal.Width' · 'Species'
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
 Median :5.800   Median :3.000   Median :4.350   Median :1.300
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
       Species
 setosa    :50
 versicolor:50
 virginica :50
```

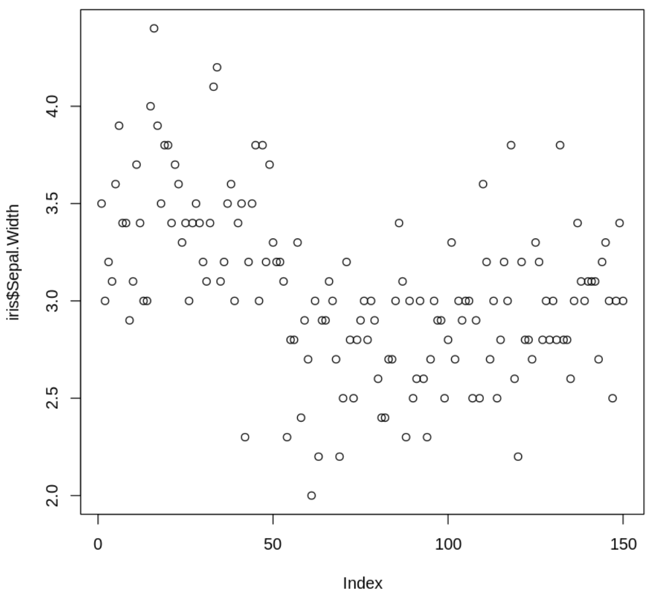
KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20
M.Sc (I.T.) Part-2 Semester IV

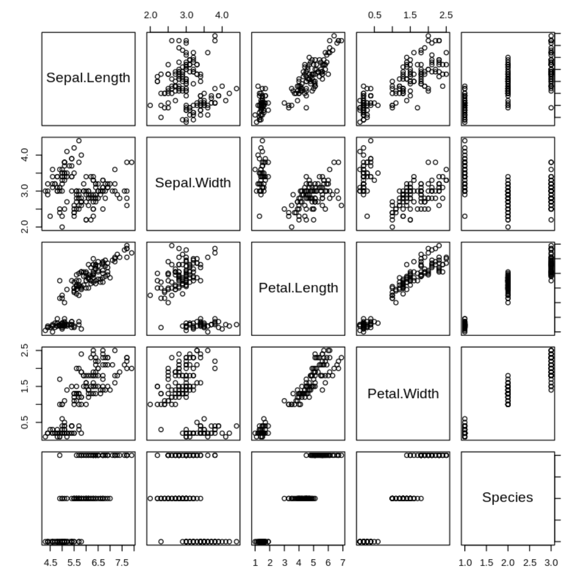
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  2.000   2.800   3.000   3.057   3.300   4.400 
```

FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALS
FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALS
FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALS
FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALS
FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALS
FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE

A matrix: 150 × 5 of type lgl

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE





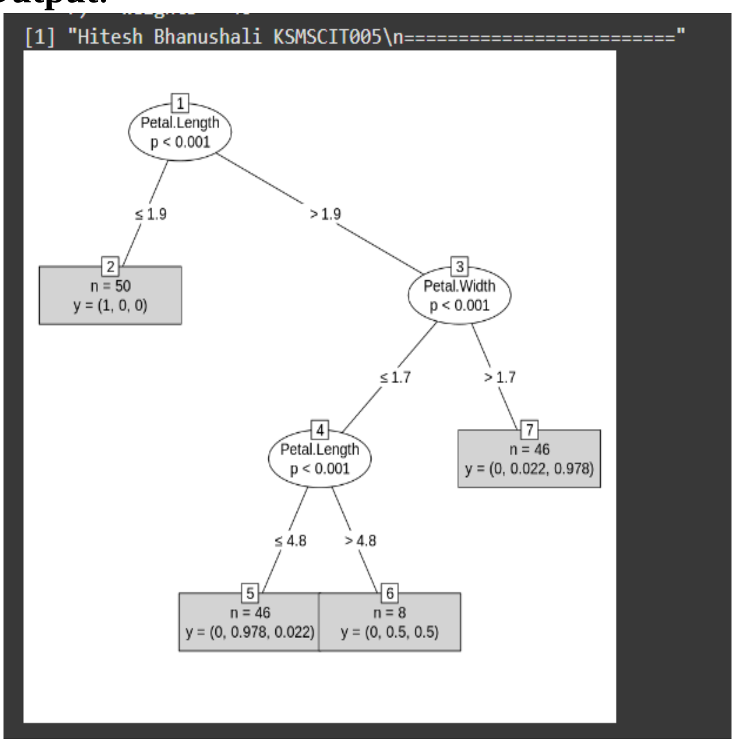
Practical 4B: Implementing Classification in R [Decision Tree Classifier]
Practical 4B: Part A

Step 01: First install the Packages.
install.packages('party')

Code:

```
library(party) # Load package party
library (datasets) # load datasets package
data (iris) # load dataset
print("Hitesh Bhanushali KSMSCIT005\n========================")
print(iris)
target = Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width
cdt <- ctree(target, iris) #Build tree
table(predict(cdt), iris$Species) # Create confusion matrix
cdt #To display decision tree rulesplot(cdt, type="simple") #Plotting of decision tree
plot(cdt, type="simple") #Plotting of decision tree
print("Hitesh Bhanushali KSMSCIT005\n========================")
```

Output:




```
Loading required package: sandwich

[1] "Hitesh Bhanushali KSMSCIT005\n========================"
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
4            4.6         3.1          1.5         0.2     setosa
5            5.0         3.6          1.4         0.2     setosa
6            5.4         3.9          1.7         0.4     setosa
7            4.6         3.4          1.4         0.3     setosa
8            5.0         3.4          1.5         0.2     setosa
9            4.4         2.9          1.4         0.2     setosa
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester IV

Practical 4C: Implementing Classification in R [Naïve Bayes]

Step 01: First install the Packages.
install.packages('e1071')

```
library("e1071")
print("Hitesh Bhanushali KSMSCIT005\n=========================")
data<- read.csv('/content/weather-nominal-weka.csv')
print(data)
print("Hitesh Bhanushali KSMSCIT005\n=========================")
weather_df = as.data.frame(data)
weather_df
print("Hitesh Bhanushali KSMSCIT005\n=========================")
# Fix: Use 'weather_df' instead of 'data-weather_df'
Naive_Bayes_Model = naiveBayes (play ~., data=weather_df)
print (Naive_Bayes_Model)
print("Hitesh Bhanushali KSMSCIT005\n=========================")
NB_Predictions = predict (Naive_Bayes_Model, weather_df)
table(NB_Predictions, weather_df$play, dnn = c('Prediction', 'Actual'))
```

```
[1] "Hitesh Bhanushali KSMSCIT005\n========================="
    outlook temperature humidity windy play
1     sunny         hot     high FALSE   no
2     sunny         hot     high  TRUE   no
3  overcast         hot     high FALSE  yes
4     rainy        mild     high FALSE  yes
5     rainy        cool   normal FALSE  yes
6     rainy        cool   normal  TRUE   no
7  overcast        cool   normal  TRUE  yes
8     sunny        mild     high FALSE   no
9     sunny        cool   normal FALSE  yes
10    rainy        mild   normal FALSE  yes
11    sunny        mild   normal  TRUE  yes
12 overcast        mild     high  TRUE  yes
13 overcast         hot   normal FALSE  yes
14    rainy        mild     high  TRUE   no
[1] "Hitesh Bhanushali KSMSCIT005\n========================="
```

A data.frame: 14 × 5

outlook <chr>	temperature <chr>	humidity <chr>	windy <lgl>	play <chr>
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes

```
[1] "Hitesh Bhanushali KSMSCIT005\n========================="

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
       no       yes 
0.3571429 0.6428571 

Conditional probabilities:
     outlook
Y      overcast     rainy     sunny
  no  0.0000000 0.4000000 0.6000000
  yes 0.4444444 0.3333333 0.2222222

     temperature
Y          cool       hot      mild
  no  0.2000000 0.4000000 0.4000000
  yes 0.3333333 0.2222222 0.4444444

     humidity
Y          high    normal
  no  0.8000000 0.2000000
  yes 0.3333333 0.6666667

     windy
Y         FALSE      TRUE
  no  0.4000000 0.6000000
```

```
[1] "Hitesh Bhanushali KSMSCIT005\n=======
          Actual
Prediction no yes
       no   4   0
       yes  1   9
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester IV

Practical 5A: Implementing Classifier in Python [SVM]

Code:

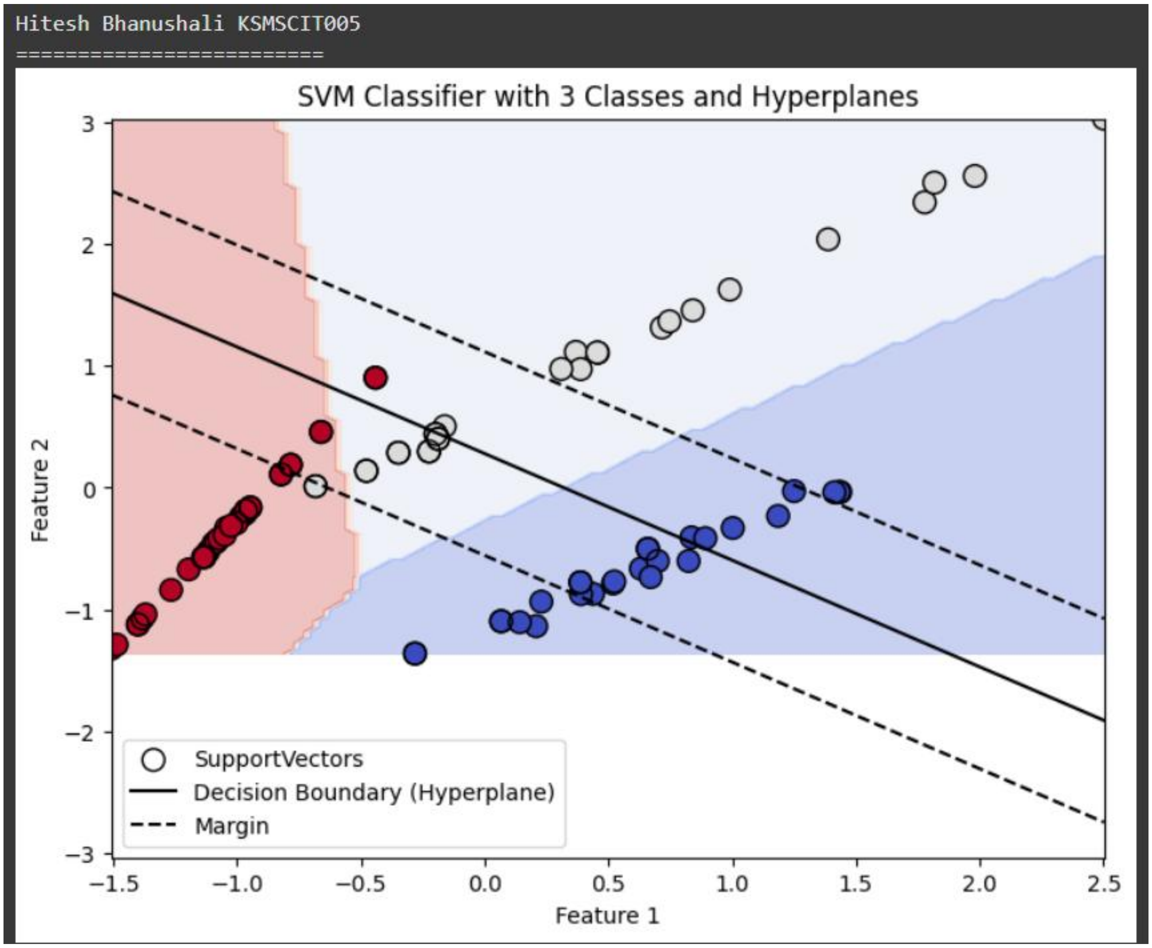
```
# Practical 5A

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
data = datasets.make_classification(n_samples=100, n_features=2, n_classes=3, n_clusters_per_class=1, random_state=42, n_informative=2, n_redundant=0, n_repeated=0)
df = pd.DataFrame(data[0], columns=['Feature 1', 'Feature 2'])
df['Target'] = data[1]
x = df.drop('Target', axis=1)
y = df[ 'Target']
print("Hitesh Bhanushali KSMSCIT005\n========================")
print(x.shape,y.shape)
```

```
Hitesh Bhanushali KSMSCIT005
========================
(100, 2) (100,)
```

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
clf = SVC (kernel='linear', decision_function_shape='ovr')
clf.fit(X_train, y_train)
test = [[1.129916,1.102361]]
Z = clf.predict(test)
print(Z)
ypred = clf.predict(X_test)
acc = clf.score (X_test, y_test)
print("Hitesh Bhanushali KSMSCIT005\n=========================")
print (f"Accuracy: {acc}")
```

```
[1]
Hitesh Bhanushali KSMSCIT005
=========================
Accuracy: 0.9666666666666667
```

```
plt.figure(figsize=(8, 6))
xx, yy = np.meshgrid (np.linspace (X_train[:, 0].min(), X_train[:, 0].max(), 100), np.linspace (X_train[:, 1].min(), X_train[:, 1].max(), 100))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
plt.contourf(xx, yy, Z, alpha=0.3, cmap=plt.cm.coolwarm)
plt.scatter (X_train[:, 0], X_train[:, 1], c=y_train, edgecolors='k', s=100, cmap=plt.cm.coolwarm)
plt.scatter (clf.support_vectors_[:, 0], clf.support_vectors_[:, 1], facecolors='none', edgecolors='k', s=100, label='SupportVectors')
w, b = clf.coef_[0], clf.intercept_[0]
xx_vals = np.linspace(X_train[:, 0].min(), X_train[:, 0].max(), 30)
yy_vals = (w[0] * xx_vals + b) / w[1] # Calculate the decision boundary line
margin = 1 / np.linalg.norm(w) # Margin distance
plt.plot(xx_vals, yy_vals, 'k-', label='Decision Boundary (Hyperplane)')
plt.plot(xx_vals, yy_vals + margin * np.linalg.norm(w) / w[1], 'k--', label='Margin')
plt.plot(xx_vals, yy_vals - margin * np.linalg.norm(w) / w[1], 'k--')
plt.title('SVM Classifier with 3 Classes and Hyperplanes')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
print("Hitesh Bhanushali KSMSCIT005\n========================")
plt.show()
```



Hitesh Bhanushali KSMSCIT005
========================
SVM Classifier with 3 Classes and Hyperplanes
Feature 2
Feature 1
SupportVectors
Decision Boundary (Hyperplane)
Margin

Practical 5B: Implementing Classifier in R Studio [SVM]

Code:

```
install.packages('e1071')
install.packages("caret")
library(e1071)   # SVM
library(caret)   # Train-test split & evaluation
data <- data.frame(
  feature1 = c(5.1, 4.9, 6.2, 5.9),
  feature2 = c(3.5, 3.0, 3.4, 3.0),
  feature3 = c(1.4, 1.4, 5.4, 5.1),
  target = as.factor(c(0, 0, 2, 1))
)
# Split into training & testing sets
set.seed(42)
trainIndex <- createDataPartition(data$target, p=0.5, list=FALSE)
trainData <- data[trainIndex, ]
testData  <- data[-trainIndex, ]
svm_model <- svm(target ~ ., data=trainData, kernel="linear", cost=1)
# Predictions & evaluation
predictions <- predict(svm_model, testData)
conf_matrix <- confusionMatrix(predictions, testData$target)
print(conf_matrix)
cat("\nHitesh Bhanushali KSMSCIT005\nAccuracy:", conf_matrix$overall["Accuracy"], "\n")
```

Output:

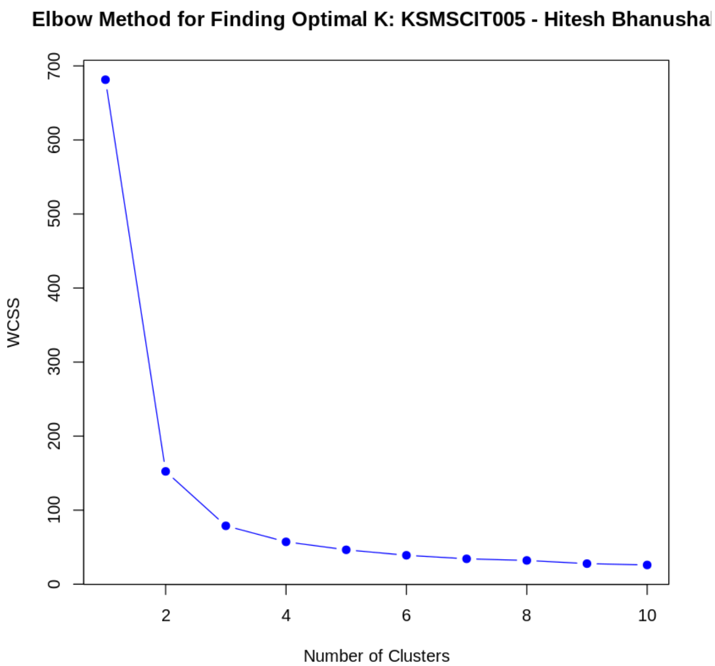
```
Confusion Matrix and Statistics

          Reference
Prediction 0 1 2
         0 1 0 0
         1 0 0 0
         2 0 0 0

Overall Statistics

               Accuracy : 1
                 95% CI : (0.025, 1)
    No Information Rate : 1
    P-Value [Acc > NIR] : 1

                  Kappa : NaN

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2
Sensitivity                 1       NA       NA
Specificity                NA        1        1
Pos Pred Value             NA       NA       NA
Neg Pred Value             NA       NA       NA
Prevalence                  1        0        0
Detection Rate              1        0        0
Detection Prevalence        1        0        0
Balanced Accuracy          NA       NA       NA

Hitesh Bhanushali KSMSCIT005
Accuracy: 1
```

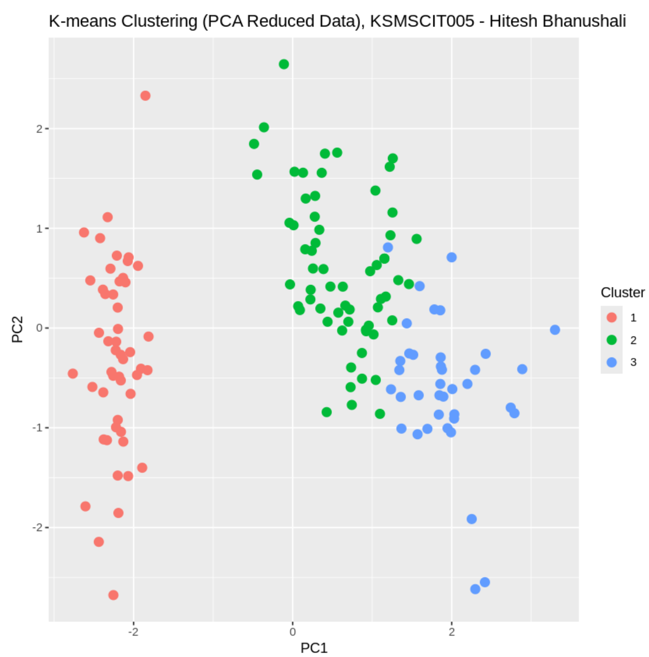
Practical 6: Implementation of K-Means in Python/R

Code:

```
print("Hitesh Bhanushali KSMSCIT005\n=========================")
library(ggplot2)
library(datasets)
data(iris)
df <- iris[, 1:4]
set.seed(123)
wcss <- vector()
for (k in 1:10) {
  wcss[k] <- sum (kmeans (df, centers = k, nstart = 10) $tot.withinss)}
  plot (1:10, wcss, type = "b", pch =19, col = "blue",
  xlab="Number of Clusters", ylab = "WCSS", main = "Elbow Method for Finding Optimal K:
KSMSCIT005 - Hitesh Bhanushali")
  set.seed(123)
kmeans_result <- kmeans (df, centers = 3, nstart = 25)
pca_result <- prcomp (df, scale. = TRUE)

df_pca <- data.frame(pca_result$x[, 1:2], Cluster = as.factor(kmeans_result$cluster))
ggplot(df_pca, aes(x = PC1, y = PC2, color = Cluster)) + geom_point (size = 3) +
labs (title="K-means Clustering (PCA Reduced Data), KSMSCIT005 - Hitesh Bhanushali")
```

Output:





Practical 7: Understating Weka

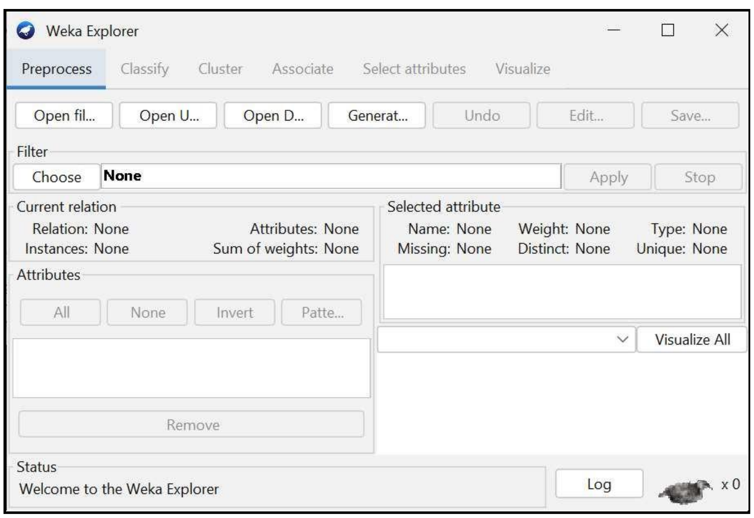
Step 01: Download and install the latest version of Weka from its Official Website, then start Weka.

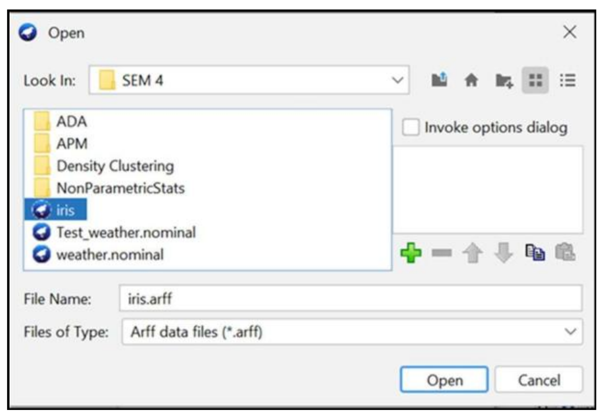
Note: To work more smoothly, you must first download and install Java VM before downloading Weka.

Step 02: Download R. A. Fisher’s Iris Flower dataset → link [Click].

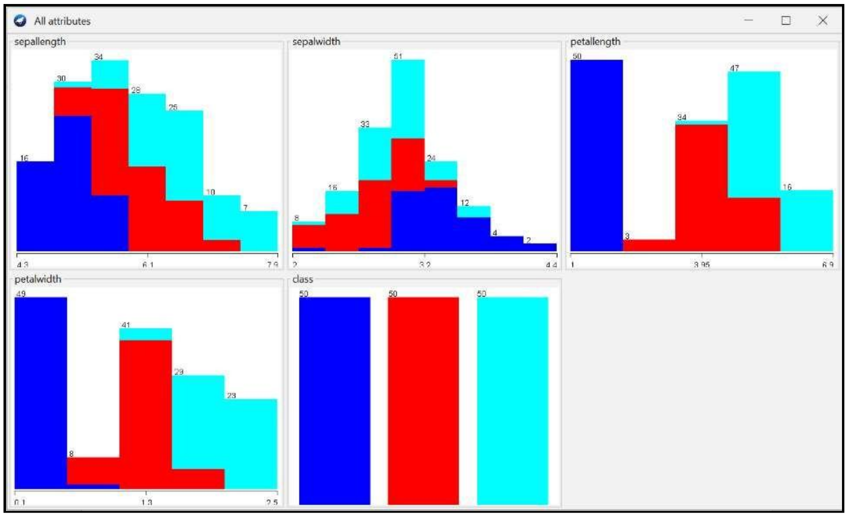
Step 03: When you start weka you will see the weka GUI in that hit Explorer tab

Step 04: Now hit open file and select ‘iris.arff File’.





Step 05: Open the file to view 'Attribute Statistics, Class Designator and Class Histogram and Expansion of class Designator'.



Histograms for all attributes of Iris dataset [see color plate]

Selected attribute
Name: petalwidth — Type: Numeric
Missing: 0 (0%) — Distinct: 22 — Unique: 2 (1%)

Statistic	Value
Minimum	0.1
Maximum	2.5
Mean	1.199
StdDev	0.763

ARFF Viewer

Program Visualization Tools Help

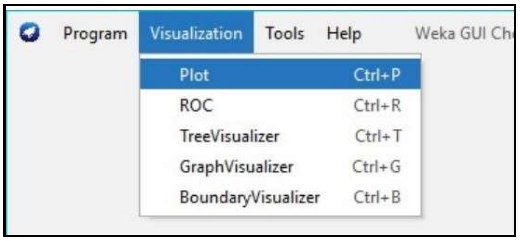
Package manager	Ctrl+U
ArffViewer	Ctrl+A
SqlViewer	Ctrl+S
Bayes net editor	Ctrl+N

File Edit View — ARFF-Viewer -

weather.nominal.arff

Relation: weather.symbolic

No.	1: outlook Nominal	2: temperature Nominal	3: humidity Nominal	4: windy Nominal	5: **play** Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

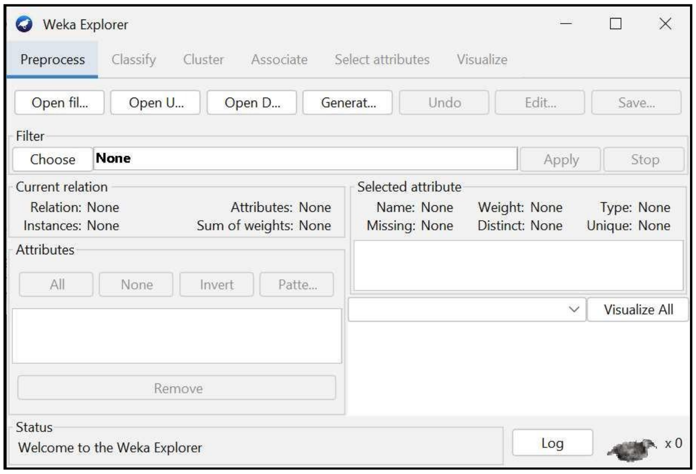


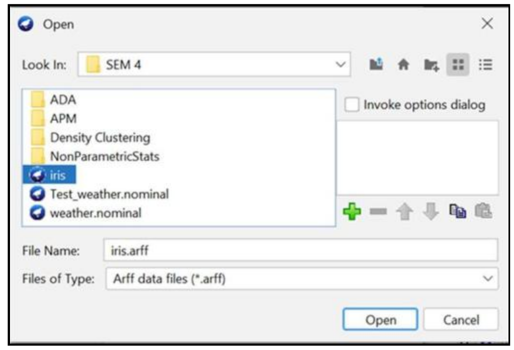
Program
Visualization
Tools
Help
Plot Ctrl+P
ROC Ctrl+R
TreeVisualizer Ctrl+T
GraphVisualizer Ctrl+G
BoundaryVisualizer Ctrl+B

Plot - D:\MSC\SEM 4\iris.arff
X: sepallength (Num)
Y: sepalwidth (Num)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: iris
4.4
3.2
2
4.3
6.1
7.9
Class colour
Iris-setosa Iris-versicolor Iris-virginica

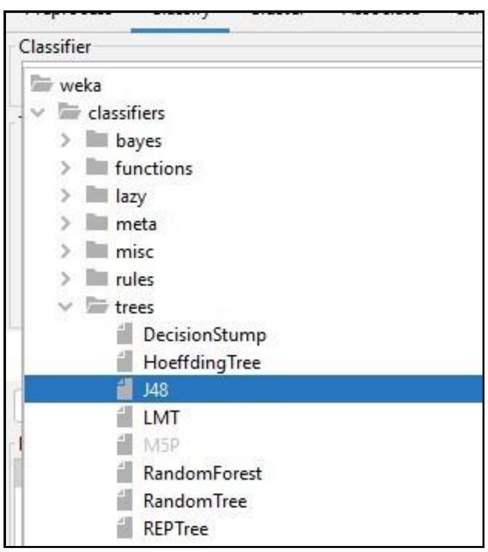
Practical 8: Implementing Classification in Weka [Decision Tree]

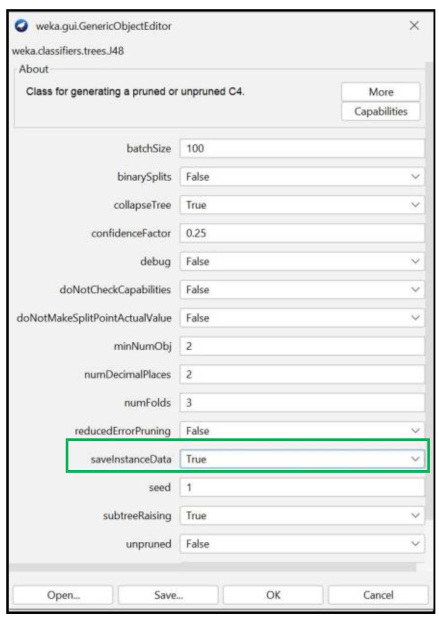
Step 01: Open Weka → Explorer → Open file → Choose 'iris.arff' file → after that move to the Classify Tab.

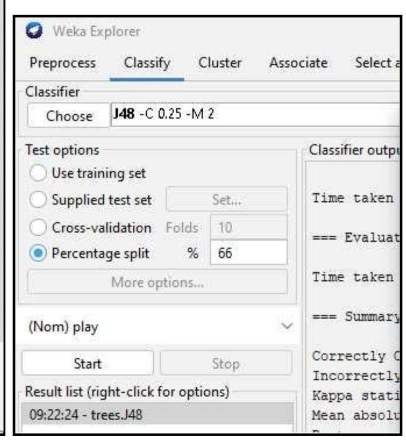


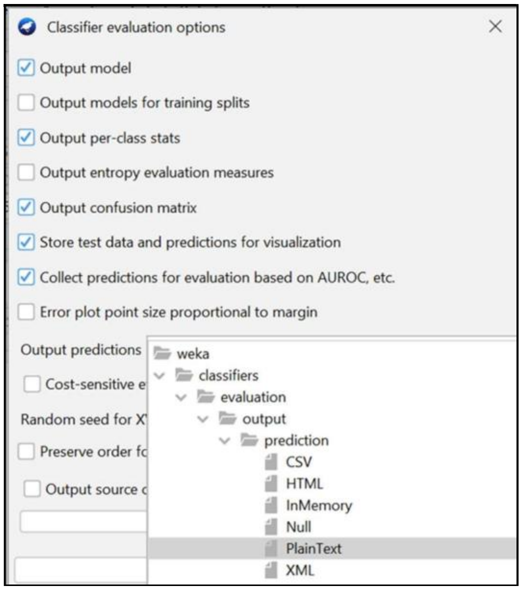


Step 02: In the Classify tab follow the path choose → trees → J48 now you will be able to see J48 in the choose textbox Click on it, set saveInstanceData to true, and confirm. Then, select Percentage split under Test options and set it to 66%, and click Start





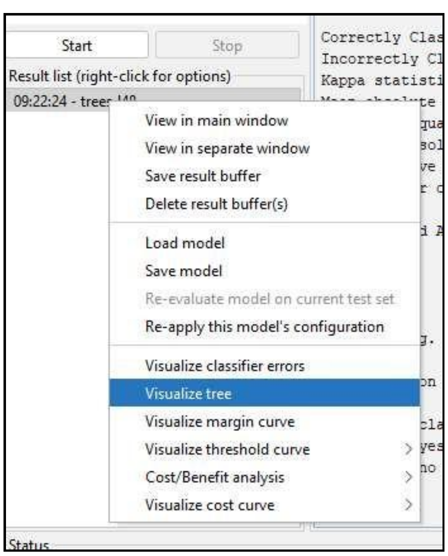


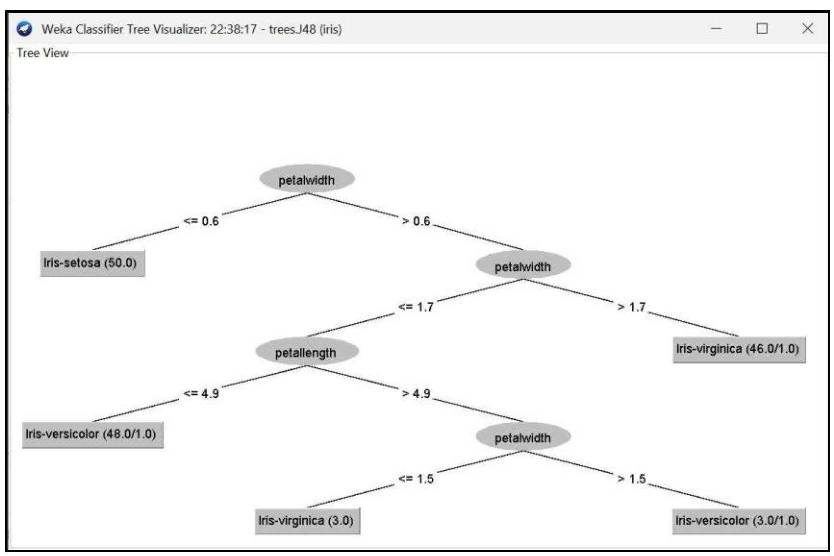


Step 03: The Classifier output box displays classification results, including accuracy statistics, as shown in the figure.

```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 15  0  0 |  a = Iris-setosa
  0 19  0 |  b = Iris-versicolor
  0  2 15 |  c = Iris-virginica
```

Step 04: Right-click on the highlighted Result list entry, then select Visualize tree to view the decision tree for the Fisher's Iris dataset. The tree is easy to read



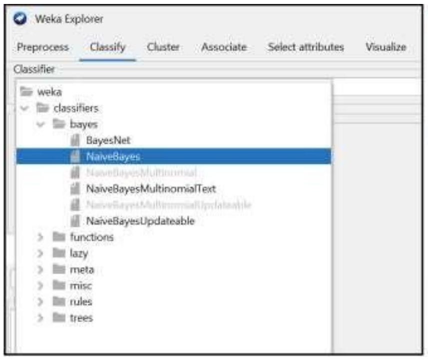


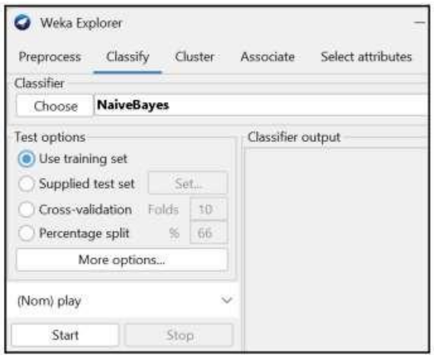
KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

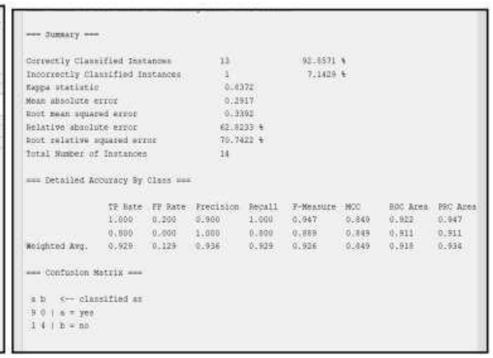
M.Sc (I.T.) Part-2 Semester IV

Practical 9: Implementing Classification in Weka [Naïve Bayes]

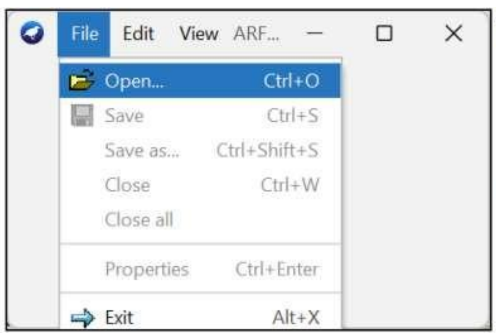
Step 01: Open Weka Explorer → Preprocess → load weather.nominal.arff → move to Classify tab → choose Naïve Bayes[it will be in bayes directory] → set classifier options via More Options → hit Use Training Set → Start → Now, you will see the results showing 13 correct, 1 incorrect [6th instance, marked [+]], with 92.8571% accuracy.

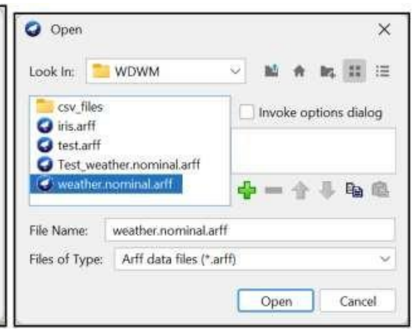




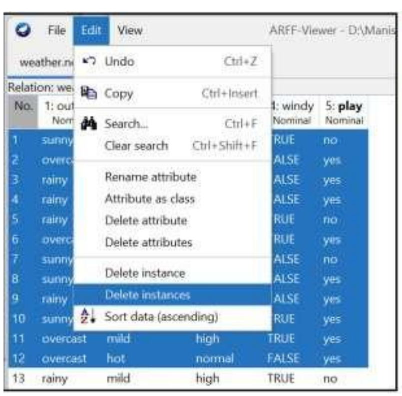


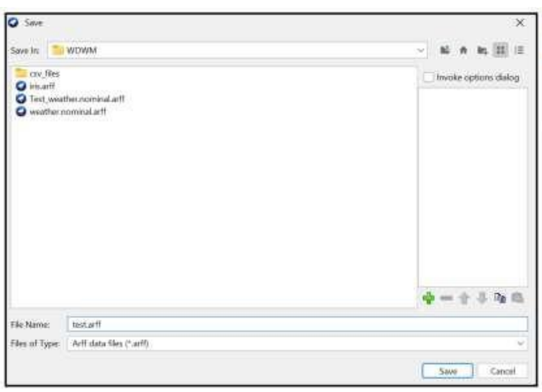
Step 02: Now will create a training dataset and for that,
Open weka GUI → Tool Tab → Arrffviewer → File → open weather.nominal.arff file.

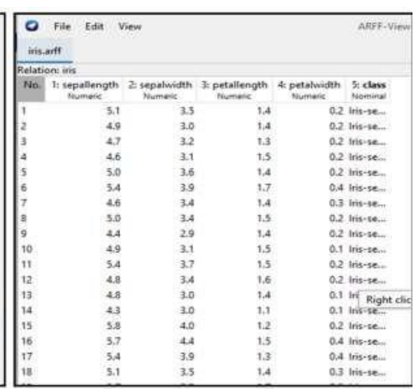




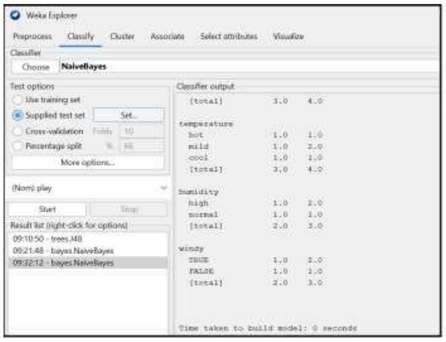
Step 03: Select all records, exclude one, delete the rest, and save as test.arff. After that Go to the Preprocess tab → Open file → iris.arff.

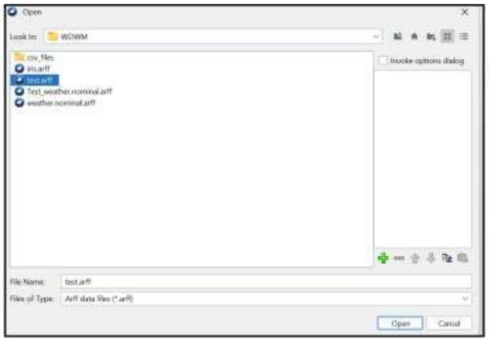


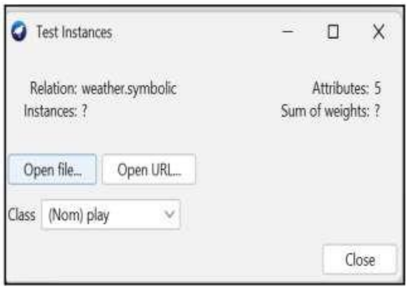


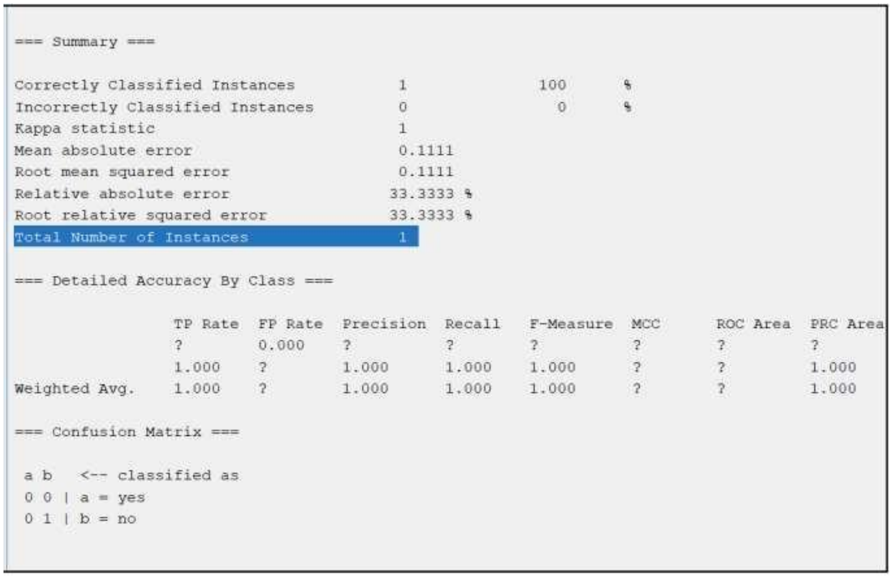


Step 04: Move to Classify tab → choose Naïve Bayes → hit Start to build the classifier, then Select Supplied test set → hit Set and open test.arff file, once the file details are displayed → hit Start again to classify the test instances. The classifier predicts an unknown instance as Play: Yes









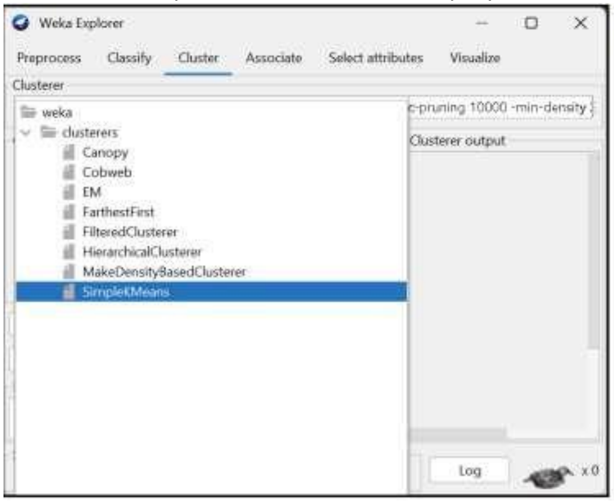
```
=== Summary ===

Correctly Classified Instances           1              100      %
Incorrectly Classified Instances         0                0      %
Kappa statistic                          1
Mean absolute error                      0.1111
Root mean squared error                  0.1111
Relative absolute error                 33.3333 %
Root relative squared error             33.3333 %
Total Number of Instances                1

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
                 ?        0.000    ?          ?        ?          ?        ?         ?
                 1.000    ?        1.000      1.000    1.000      ?        ?         1.000
Weighted Avg.    1.000    ?        1.000      1.000    1.000      ?        ?         1.000

=== Confusion Matrix ===

 a b   <-- classified as
 0 0 | a = yes
 0 1 | b = no
```

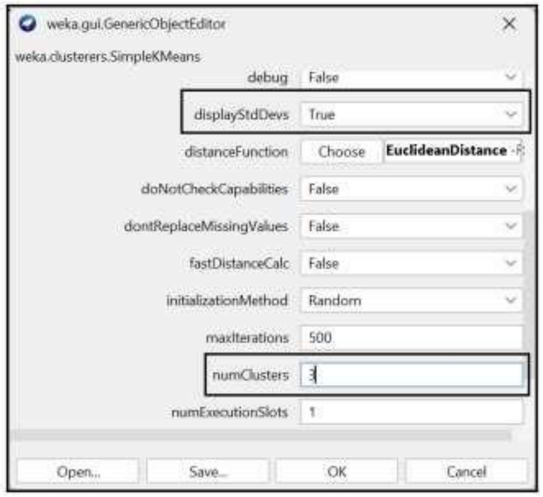
KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester IV

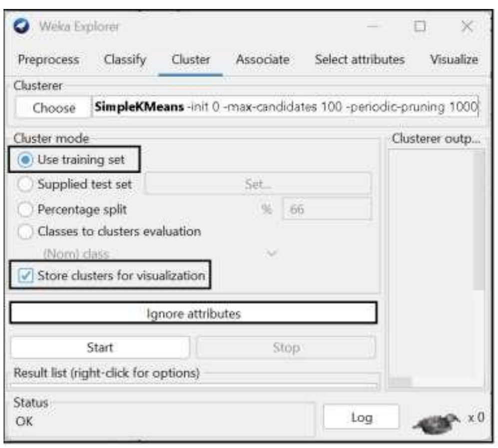
Practical 10: Implementing Clustering with Weka

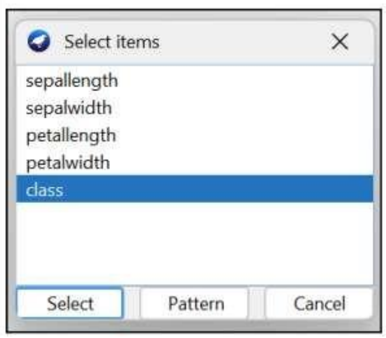
Step 01: Open Weka Explorer → Preprocess → load iris.arff file → move to cluster tab → choose simple k-means → click on simple kmeans and set 'displaystdDevs as True and numClusters as 3'

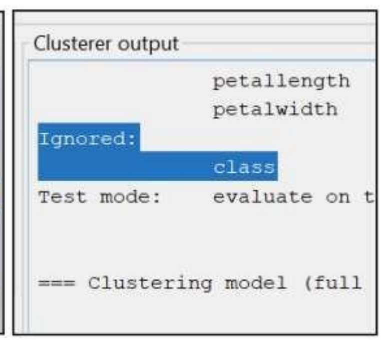




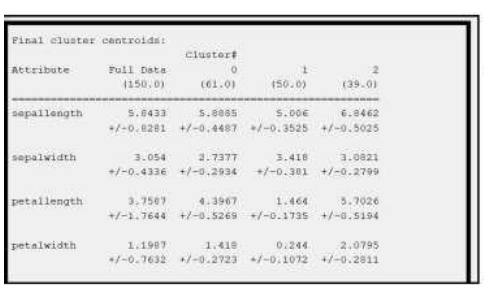
Step 02: select Use training set → tick store clusters for visualization → hit Ignore attributes → select Class → hit start.



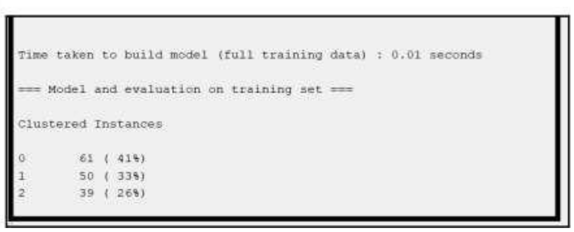




Step 03: In the Cluster output we can see that the K-Means algorithm forms three clusters with 61, 50, and 39 instances.

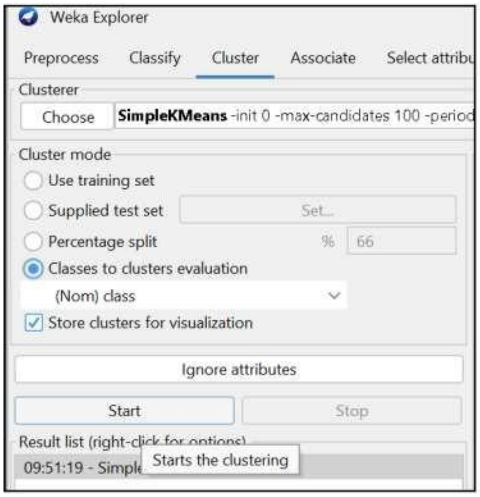


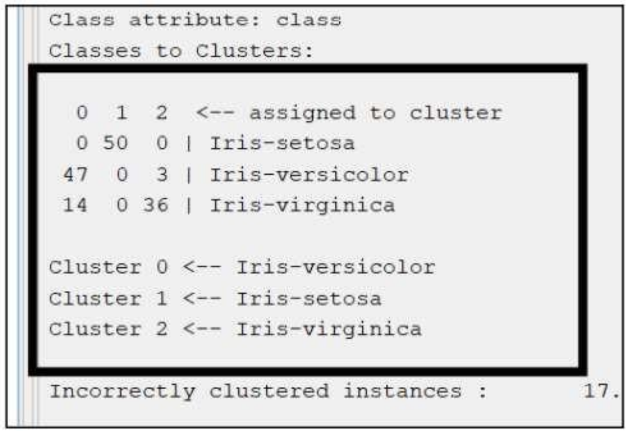
```
Final cluster centroids:
                         Cluster#
Attribute    Full Data          0          1          2
               (150.0)     (61.0)     (50.0)     (39.0)
=======================================================
sepallength     5.8433     5.8885      5.006     6.8462
            +/-0.8281  +/-0.4487  +/-0.3525  +/-0.5025

sepalwidth       3.054     2.7377      3.418     3.0821
            +/-0.4336  +/-0.2934   +/-0.381  +/-0.2799

petallength     3.7587     4.3967      1.464     5.7026
            +/-1.7644  +/-0.5269  +/-0.1735  +/-0.5194

petalwidth      1.1987      1.418      0.244     2.0795
            +/-0.7632  +/-0.2723  +/-0.1072  +/-0.2811
```



```
Time taken to build model (full training data) : 0.01 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       61 ( 41%)
1       50 ( 33%)
2       39 ( 26%)
```

Step 04: To compare results with actual clusters, select Classes to Cluster Evaluation in Cluster mode and re-run the algorithm.




```
Class attribute: class
Classes to Clusters:

  0  1  2  <-- assigned to cluster
  0 50  0 | Iris-setosa
 47  0  3 | Iris-versicolor
 14  0 36 | Iris-virginica

Cluster 0 <-- Iris-versicolor
Cluster 1 <-- Iris-setosa
Cluster 2 <-- Iris-virginica

Incorrectly clustered instances :      17.
```

Step 05: You can also visualize the cluster. Clusters can be visualized using any input attribute. Clusters plotted with Petal Length and Petal Width. Increase Jitter to view all samples.