PRACTICAL JOURNAL

in

DATA MANAGEMENT AND DATA WAREHOUSING

Submitted by

KSMSCIT033 ASHISH ANIL SHENDE

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. **ASHISH ANIL SHENDE** for M.Sc. (IT) Part- II Semester- III, Seat No: **KSMSCIT033** 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.

```
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("Ashish Shende - KSMSCIT033")
print("--------------------------")
print("Outliers: \n", outliers)
```

```
Ashish Shende - KSMSCIT033
--------------------------
Outliers:
 Empty DataFrame
Columns: [Value]
Index: []
```

```
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("Ashish Shende - KSMSCIT033")
print("--------------------------")
print(outliers)
```

```
[25, 30, 28, 200, 22, 40, 29, 31, 27, 33]
Ashish Shende - KSMSCIT033
--------------------------
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:

```
import pandas as pd
print("KSMSCIT033 - Ashish Shende")
print("---------------------------")

df = pd.read_csv("/content/train.csv")
print(df.head())
```

```
KSMSCIT033 - Ashish Shende
---------------------------
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

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

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

Calculate Descriptive Summary Statistics

```
print("KSMSCIT033 - Ashish Shende")
print("---------------------------")
print(df.describe())
```

```
KSMSCIT033 - Ashish Shende
---------------------------
       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("KSMSCIT033 - Ashish Shende")
print("---------------------------")
print(df.describe(include='all'))
```

```
KSMSCIT033 - Ashish Shende
---------------------------
        PassengerId    Survived      Pclass                 Name   Sex  \
count    891.000000  891.000000  891.000000                  891   891
unique          NaN         NaN         NaN                  891     2
top             NaN         NaN         NaN  Dooley, Mr. Patrick  male
freq            NaN         NaN         NaN                    1   577
mean     446.000000    0.383838    2.308642                  NaN   NaN
std      257.353842    0.486592    0.836071                  NaN   NaN
min        1.000000    0.000000    1.000000                  NaN   NaN
25%      223.500000    0.000000    2.000000                  NaN   NaN
50%      446.000000    0.000000    3.000000                  NaN   NaN
75%      668.500000    1.000000    3.000000                  NaN   NaN
max      891.000000    1.000000    3.000000                  NaN   NaN

               Age       SibSp       Parch  Ticket        Fare Cabin Embarked
count   714.000000  891.000000  891.000000     891  891.000000   204      889
unique         NaN         NaN         NaN     681         NaN   147        3
top            NaN         NaN         NaN  347082         NaN    G6        S
freq           NaN         NaN         NaN       7         NaN     4      644
mean     29.699118    0.523008    0.381594     NaN   32.204208   NaN      NaN
std      14.526497    1.102743    0.806057     NaN   49.693429   NaN      NaN
min       0.420000    0.000000    0.000000     NaN    0.000000   NaN      NaN
25%      20.125000    0.000000    0.000000     NaN    7.910400   NaN      NaN
50%      28.000000    0.000000    0.000000     NaN   14.454200   NaN      NaN
75%      38.000000    1.000000    0.000000     NaN   31.000000   NaN      NaN
max      80.000000    8.000000    6.000000     NaN  512.329200   NaN      NaN
```

Identify potential features and target variables

```
print("KSMSCIT033 - Ashish Shende")
print("---------------------------")
x = df.drop(["Survived"], axis=1)
y = df ["Survived"]
print("Features : \n",x.head(3))
print("Target : \n",y.head(3))

print("---------------------------")
```

```
KSMSCIT033 - Ashish Shende
---------------------------
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

      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
Target :
 0    0
1    1
2    1
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.

```
import matplotlib.pyplot as plt
# create histogram for numeric data
df.hist(figsize=(7,7))
# show plot
print("KSMSCIT033 - Ashish Shende")
print("--------------------------")
plt.show()
```

Column Chart

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

```
import matplotlib.pyplot as plt
df.plot.bar()
# plot between 2 attributes
plt.bar(df['Age'], df ['Pclass'])
plt.xlabel("Age")
plt.ylabel("Pclass")
print("KSMSCIT033 - Ashish Shende")
plt.show()
```

Scatter Plot

```
plt.scatter(df["Pclass"], df["Age"])
plt.xlabel("Pclass")
plt.ylabel("Age")
plt.title("Scatter Plot")
print("KSMSCIT033 - Ashish Shende")
plt.show()
```

Box Plot

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(x='Pclass', y='Age', data=df)
print("Glory Lithiyal - KSMSCIT008")
print("---------------------------")
plt.show()
```

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(x='Pclass', y='Age', data=df)
print("KSMSCIT033 - Ashish Shende")
print("---------------------------")
plt.show()
```

```
KSMSCIT033 - Ashish Shende
---------------------------
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

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
# Import dataset
df = pd.read_csv("/content/iris.csv")
print("KSMSCIT033 - Ashish Shende")
print("-------------------")
print(df['species'].unique())
```

```
KSMSCIT033 - Ashish Shende
-------------------
['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 object knows
# how to understand word labels.
label_encoder = preprocessing. LabelEncoder()
# Encode labels in column 'species'.
df['species']= label_encoder.fit_transform(df['species'])
print("KSMSCIT033 - Ashish Shende")
print("-------------------")
print(df['species'].unique())
```

```
KSMSCIT033 - Ashish Shende
-------------------
[0 1 2]
```

```python
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.preprocessing import Binarizer
# Load dataset
data = load_iris(as_frame=True)
df = data.frame
print("KSMSCIT033 - Ashish Shende")
print("-------------------")
print(df['sepal length (cm)'])
```

```
KSMSCIT033 - Ashish Shende
-------------------
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

```python
binarizer = Binarizer(threshold=5.0)
df['sepal length (cm) binary'] = binarizer.fit_transform(df[['sepal length (cm)']])
print("KSMSCIT033 - Ashish Shende")
print("-------------------")
print(df['sepal length (cm) binary'])
```

```
KSMSCIT033 - Ashish Shende
-------------------
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
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

Practical 2A: Implementation of Decision Tree using Information Gain

```
import numpy as np
import pandas as pd
print("KSMSCIT033 - Ashish Shende")
print("---------------------------")
data = pd.read_csv('/content/iris.csv')
print(data.head())
```

```
KSMSCIT033 - Ashish Shende
---------------------------
   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
```

```
x = data.drop('species', axis=1)
y = data['species']
print("KSMSCIT033 - Ashish Shende")
print("---------------------------")
print("Features : \n",x.head(5))
print("Target : \n",y.head(5))
```

```
KSMSCIT033 - Ashish Shende
---------------------------
Features :
    sepal_length  sepal_width  petal_length  petal_width
0           5.1          3.5           1.4          0.2
1           4.9          3.0           1.4          0.2
2           4.7          3.2           1.3          0.2
3           4.6          3.1           1.5          0.2
4           5.0          3.6           1.4          0.2
Target :
 0    setosa
1    setosa
2    setosa
3    setosa
4    setosa
Name: species, dtype: object
```

```
# Split dataset into Training and Testing

from sklearn.model_selection import train_test_split
print("KSMSCIT033 - Ashish Shende")
print("---------------------------")
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
print("Training Data")
print(x_train)
print(y_train)
print("Testing Data")
print(x_test)
print(y_test)
```

```
KSMSCIT033 - Ashish Shende
--------------------------
Training Data
     sepal_length  sepal_width  petal_length  petal_width
22            4.6          3.6           1.0          0.2
15            5.7          4.4           1.5          0.4
65            6.7          3.1           4.4          1.4
11            4.8          3.4           1.6          0.2
42            4.4          3.2           1.3          0.2
..            ...          ...           ...          ...
71            6.1          2.8           4.0          1.3
106           4.9          2.5           4.5          1.7
14            5.8          4.0           1.2          0.2
92            5.8          2.6           4.0          1.2
102           7.1          3.0           5.9          2.1

[120 rows x 4 columns]
```

```
training Features:
 22         Iris-setosa
15         Iris-setosa
65     Iris-versicolor
11         Iris-setosa
42         Iris-setosa
            ...
71     Iris-versicolor
106     Iris-virginica
14         Iris-setosa
92     Iris-versicolor
102     Iris-virginica
Name: Species, Length: 120, dtype: object
```

```
Testing Data
     sepal_length  sepal_width  petal_length  petal_width
73            6.1          2.8           4.7          1.2
18            5.7          3.8           1.7          0.3
118           7.7          2.6           6.9          2.3
78            6.0          2.9           4.5          1.5
76            6.8          2.8           4.8          1.4
31            5.4          3.4           1.5          0.4
64            5.6          2.9           3.6          1.3
141           6.9          3.1           5.1          2.3
68            6.2          2.2           4.5          1.5
82            5.8          2.7           3.9          1.2
110           6.5          3.2           5.1          2.0
12            4.8          3.0           1.4          0.1
36            5.5          3.5           1.3          0.2
9             4.9          3.1           1.5          0.1
19            5.1          3.8           1.5          0.3
56            6.3          3.3           4.7          1.6
104           6.5          3.0           5.8          2.2
69            5.6          2.5           3.9          1.1
55            5.7          2.8           4.5          1.3
132           6.4          2.8           5.6          2.2
29            4.7          3.2           1.6          0.2
127           6.1          3.0           4.9          1.8
26            5.0          3.4           1.6          0.4
128           6.4          2.8           5.6          2.1
131           7.9          3.8           6.4          2.0
145           6.7          3.0           5.2          2.3
108           6.7          2.5           5.8          1.8
143           6.8          3.2           5.9          2.3
45            4.8          3.0           1.4          0.3
30            4.8          3.1           1.6          0.2
```

```python
# Model Creation and Training
from sklearn.tree import DecisionTreeClassifier
print("KSMSCIT033 - Ashish Shende")
print("---------------------------")
model = DecisionTreeClassifier()
model.fit(x_train, y_train)
```

```
KSMSCIT033 - Ashish Shende
---------------------------
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
y_pred = model.predict(x_test)
print("KSMSCIT033 - Ashish Shende")
print("---------------------------")
print("Predicted Values : \n",y_pred)
```

```
KSMSCIT033 - Ashish Shende
---------------------------
Predicted Values :
 ['versicolor' 'setosa' 'virginica' 'versicolor' 'versicolor' 'setosa'
 'versicolor' 'virginica' 'versicolor' 'versicolor' 'virginica' 'setosa'
 'setosa' 'setosa' 'setosa' 'versicolor' 'virginica' 'versicolor'
 'versicolor' 'virginica' 'setosa' 'virginica' 'setosa' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'setosa' 'setosa']
```

```
# Calculate Accuracy of Model
from sklearn.metrics import accuracy_score
print("KSMSCIT033 - Ashish Shende")
print("---------------------------")
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
KSMSCIT033 - Ashish Shende
---------------------------
Accuracy: 1.0
```

```
# Plot Decision Tree
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
plt.figure(figsize=(10, 6))
plot_tree (model, filled=True, feature_names=x.columns, class_names=model.classes_)
plt.title("KSMSCIT033 - Ashish Shende")
plt.show()
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

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

```
print("KSMSCIT033 - Ashish Shende")
print("---------------------------")
from sklearn.datasets import load_iris
data = load_iris()

x = data.data
y = data.target

print("X: \n",x)
print("y: \n",y)
```

```
KSMSCIT033 - Ashish Shende
---------------------------
X:
 [[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
```

```
y:
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

```
print("KSMSCIT033 - Ashish ShendeL")
print("---------------------------")
from sklearn.tree import DecisionTreeClassifier, export_text, plot_tree
model=  DecisionTreeClassifier (criterion='gini', random_state=42)
model.fit(x, y)
```

```
KSMSCIT033 - Ashish ShendeL
---------------------------
▾ DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

```
test_data = [[6.1,3.4,9,1.8]]
prediction = model.predict(test_data)
print("KSMSCIT033 - Ashish Shende")
print("---------------------------")
print("Predicted :", prediction)
```

```
KSMSCIT033 - Ashish Shende
---------------------------
Predicted : [2]
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
plot_tree (model, filled=True, feature_names=data.feature_names, class_names=data.target_names)
plt.title("KSMSCIT033 - Ashish Shende")
plt.show()
```

```
tree_rules = export_text(model, feature_names=data.feature_names)
print("KSMSCIT033 - Ashish Shende")
print("---------------------------")
print(tree_rules)
```

```
KSMSCIT033 - Ashish Shende
--------------------------
|--- 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
|   |   |   |   |--- petal length (cm) <= 5.45
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- petal length (cm) >  5.45
|   |   |   |   |   |--- class: 2
|   |--- petal width (cm) >  1.75
|   |   |--- petal length (cm) <= 4.85
|   |   |   |--- sepal length (cm) <= 5.95
|   |   |   |   |--- class: 1
|   |   |   |--- sepal length (cm) >  5.95
|   |   |   |   |--- class: 2
|   |   |--- petal length (cm) >  4.85
|   |   |   |--- class: 2
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

Practical 2C: Implementing of Gini Index in Python

```
parent_node = [50, 30, 20] # Class A: 50, Class B: 30, Class C: 20
child_1= [30, 20, 10] # Class A: 30, Class B: 20, Class C: 10
child_2 = [20, 10, 10] # Class A: 20, Class B: 10, Class C: 10

print("KSMSCIT033 - Ashish Shende")
print("---------------------------")
print("Parent_Node:", parent_node)
print("Child_1", child_1)
print("Child_2", child_2)
```

```
KSMSCIT033 - Ashish Shende
---------------------------
Parent_Node: [50, 30, 20]
Child_1 [30, 20, 10]
Child_2 [20, 10, 10]
```

```
def gini_index(classes):
  total = sum(classes)
  gini = 1
  for c in classes:
    gini -= (c/total)**2
  return gini

print("KSMSCIT033 - Ashish Shende")
print("---------------------------")

gini_parent = gini_index(parent_node)
print("Parent Gini Index:", gini_parent)
```

```
KSMSCIT033 - Ashish Shende
---------------------------
Parent Gini Index: 0.62
```

```
def weighted_gini (children):
  total = sum(sum(child) for child in children)
  weighted_gini = 0
  for child in children:
    weighted_gini += sum(child) / total * gini_index(child)
  return weighted_gini

gini_split = weighted_gini ([child_1, child_2])
print("KSMSCIT033 - Ashish Shende")
print("---------------------------")
print("weighted_gini: \n", gini_split)
```

```
KSMSCIT033 - Ashish Shende
---------------------------
weighted_gini:
 0.6166666666666667
```

```
def gini_gain (parent, children):
  gini_parent = gini_index(parent)
  gini_split = weighted_gini (children)
  return gini_parent - gini_split

gain = gini_gain (parent_node, [child_1, child_2])
print("KSMSCIT033 - Ashish Shende")
print("---------------------------")
print("gini_gain: \n", gain)
```

```
KSMSCIT033 - Ashish Shende
---------------------------
gini_gain:
 0.0033333333333332993
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

Practical 2D: Implementation of Decision Tree using Information gain

```python
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, export_text, plot_tree
import matplotlib.pyplot as plt

# Step 1: Load the dataset
# Using the Iris dataset as an example
data = load_iris()
X = data.data  # Features
y = data.target  # Labels

# Step 2: Train a Decision Tree Classifier with Information Gain (Entropy)
clf = DecisionTreeClassifier(criterion='entropy', max_depth=4, random_state=42)
clf.fit(X, y)

# Step 3: Visualize the Decision Tree
plt.figure(figsize=(12, 8))
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)")
print("KSMSCIT033 - Ashish Shende")
print("========================")
plt.show()

# Step 4: Print the Decision Tree in Text Format
tree_rules = export_text(clf, feature_names=data.feature_names)
print("KSMSCIT033 - Ashish Shende")
print("========================")
print(tree_rules)
```

```
KSMSCIT033 - Ashish Shende
========================
```

```
KSMSCIT033 - Ashish Shende
========================
|--- 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
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

Practical 2E: Implementation of Information gain

```python
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
parent_node = [50, 30, 20]  # Class A: 50, Class B: 30, Class C: 20
child_1 = [30, 20, 10]      # Class A: 30, Class B: 20, Class C: 10
child_2 = [20, 10, 10]      # Class A: 20, Class B: 10, Class C: 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("KSMSCIT033 - Ashish Shende")
print("========================")
print(f"Entropy (Parent Node): {parent_entropy:.4f}")
print(f"Weighted Entropy (After Split): {weighted_entropy:.4f}")
print(f"Information Gain: {gain:.4f}")
print("========================")
```

```
KSMSCIT033 - Ashish Shende
========================
Entropy (Parent Node): 1.4855
Weighted Entropy (After Split): 2.9591
Information Gain: 0.0100
========================
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

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)

Where:

- **Spam** is labeled as 1.
- **Ham** is labeled as 0.

```
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("KSMSCIT033 - Ashish Shende")
print("========================")
x = np.array([
[1, 1, 1, 0, 0], [1, 1, 0, 0, 0], [1, 0, 0, 0, 0], [0, 0, 0, 1, 1], [0, 0, 0, 1, 1], [0, 0, 0, 1, 1]])
y = np.array([1, 1, 1, 0, 0, 0])
#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)
#Step 03: Initialize the Naive Bayes classifier
model = BernoulliNB()
#Step 04: Train thr the classifier
model.fit(x_train, y_train)
#Step 05: Make the predictions on the test set
y_pred = model.predict(x_test)
#Step 06: Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
# Example of predicting the class for a new email
new_email = np.array([[0, 0, 0, 1, 1]]) # New email with words "free" and "money"
new_prediction = model.predict(new_email)
print(f'Prediction class for the new email (0=Ham, 1=Spam): {new_prediction[0]}')
```

When we use testing data as [1,1,0,0,0]

```
KSMSCIT033 - Ashish Shende
========================
Accuracy: 100.00%
Prediction class for the new email (0=Ham, 1=Spam): 1
```

When we use testing data as [0,0,0,1,1]

```
KSMSCIT033 - Ashish Shende
========================
Accuracy: 100.00%
Prediction class for the new email (0=Ham, 1=Spam): 0
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

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

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = GaussianNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("KSMSCIT033 - Ashish Shende")
print("========================")
print('Predicted Program Output:', y_pred)
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy', accuracy)
```

```
KSMSCIT033 - Ashish Shende
========================
Predicted Program 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
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

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

```
library(datasets)
data(iris)
print("KSMSCIT033 - Ashish Shende")
print("---------------------------")
print(iris)
```

```
[1] "Glory Lithiyal - KSMSCIT008"
[1] "---------------------------"
    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
```

```
names(iris)
print("----------------------------")
print("KSMSCIT033 - Ashish Shende")
print("----------------------------")
summary(iris)
```

```
'Sepal.Length' · 'Sepal.Width' · 'Petal.Length' · 'Petal.Width' · 'Species'
[1] "----------------------------"
[1] "Glory Lithiyal-KSMSCIT008"
[1] "----------------------------"
  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  
```

```
print("KSMSCIT033 - Ashish Shende")
print("----------------------------")
summary(iris$Sepal.Width)
```

```
[1] "Glory Lithiyal-KSMSCIT008"
[1] "----------------------------"
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  2.000   2.800   3.000   3.057   3.300   4.400 
```

```
print("KSMSCIT033 - Ashish Shende")
print("----------------------------")
is.na(iris$Sepal.Width)
is.na(iris)
```

```
[1] "Glory Lithiyal-KSMSCIT008"
[1] "----------------------------"
FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE ·
FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE ·
FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE ·
FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · FALSE · 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
FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE

```
print("KSMSCIT033 - Ashish Shende")
print("----------------------------")
length(unique(iris$Sepal.Width))
```

```
[1] "Glory Lithiyal-KSMSCIT008"
[1] "----------------------------"
23
```

```
print("KSMSCIT033 - Ashish Shende")
print("----------------------------")
plot(iris$Sepal.Width)
plot(iris)
```

```
print("KSMSCIT033 - Ashish Shende")
print("----------------------------")
plot(iris$Petal.Width, iris$Petal.Length)
```

```
[1] "KSMSCIT033 - Ashish Shende"
[1] "----------------------------"
```

```
[1] "KSMSCIT033 - Ashish Shende"
[1] "----------------------------"
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

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

Practical 4B: Part A

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

```
install.packages('party')

Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)

also installing the dependencies 'TH.data', 'libcoin', 'matrixStats', 'multcomp', 'mvtnorm', 'modeltools', 'strucchange', 'coin', 'zoo', 'sandwich'
```

```
library(party) # Load package party
library (datasets) # load datasets package
data (iris) # load dataset
print("KSMSCIT033 - Ashish Shende")
print("========================")
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
```

```
[1] "KSMSCIT033 - Ashish Shende"
[1] "========================"
    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
```

```
            setosa versicolor virginica
 setosa         50          0         0
 versicolor      0         49         5
 virginica       0          1        45

        Conditional inference tree with 4 terminal nodes

Response:  Species
Inputs:  Sepal.Length, Sepal.Width, Petal.Length, Petal.Width
Number of observations:  150

1) Petal.Length <= 1.9; criterion = 1, statistic = 140.264
  2)*  weights = 50
1) Petal.Length > 1.9
  3) Petal.Width <= 1.7; criterion = 1, statistic = 67.894
    4) Petal.Length <= 4.8; criterion = 0.999, statistic = 13.865
      5)*  weights = 46
    4) Petal.Length > 4.8
      6)*  weights = 8
  3) Petal.Width > 1.7
    7)*  weights = 46
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

Practical 4B [Part B]: Training and Testing of the Dataset

```
print("KSMSCIT033 - Ashish Shende")
print("========================")
library(party)
library(datasets)
indexes = sample(150,90)
iris_train = iris [indexes,]
iris_test = iris[-indexes,]
target = Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width
cdt <- ctree(target, iris_train)
table(predict(cdt, newdata = iris_test), iris_test$Species)
summary(iris_test)
plot(cdt, type="simple", main="Decision Tree - KSMSCIT033 - Ashish Shende")
```

```
[1] "KSMSCIT033 - Ashish Shende"
[1] "========================"

             setosa versicolor virginica
  setosa         19          0         0
  versicolor      0         18         2
  virginica       0          1        20
  Sepal.Length    Sepal.Width    Petal.Length   Petal.Width          Species
 Min.   :4.400   Min.   :2.00   Min.   :1.00   Min.   :0.100   setosa    :19
 1st Qu.:5.275   1st Qu.:2.80   1st Qu.:1.60   1st Qu.:0.375   versicolor:19
 Median :5.900   Median :3.00   Median :4.40   Median :1.350   virginica :22
 Mean   :5.920   Mean   :3.05   Mean   :3.88   Mean   :1.240
 3rd Qu.:6.600   3rd Qu.:3.40   3rd Qu.:5.35   3rd Qu.:1.800
 Max.   :7.700   Max.   :3.90   Max.   :6.70   Max.   :2.500
```

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')

```
install.packages('e1071')
```

```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)

also installing the dependency 'proxy'
```

```
print("KSMSCIT033 - Ashish Shende")
print("========================")
library ('e1071')
data<-read.csv('/content/weather-nominal-weka.csv')
print(data)
print("KSMSCIT033 - Ashish Shende")
print("========================")
weather_df = as.data.frame(data)
weather_df
print("KSMSCIT033 - Ashish Shende")
print("========================")
Naive_Bayes_Model=naiveBayes(play ~., data=weather_df)
print(Naive_Bayes_Model)
print("KSMSCIT033 - Ashish Shende")
print("========================")
NB_Predictions = predict(Naive_Bayes_Model, weather_df)
table(NB_Predictions, weather_df$play, dnn = c('Prediction','Actual'))
```

```
[1] "KSMSCIT033 - Ashish Shende"
[1] "========================"
    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] "KSMSCIT033 - Ashish Shende"
[1] "========================"
```

A data.frame: 14 × 5

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

```
[1] "KSMSCIT033 - Ashish Shende"
[1] "========================"

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
  yes 0.6666667 0.3333333
```

```
[1] "KSMSCIT033 - Ashish Shende"
[1] "========================"
          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]

```
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("KSMSCIT033 - Ashish Shende")
print("========================")
print(x.shape,y.shape)
```

```
KSMSCIT033 - Ashish Shende
========================
(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("KSMSCIT033 - Ashish Shende")
print("========================")
print(f"Accuracy: {acc}")
```

```
[1]
KSMSCIT033 - Ashish Shende
========================
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("KSMSCIT033 - Ashish Shende")
print("========================")
plt.show()
```

```
KSMSCIT033 - Ashish Shende
========================
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

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

```r
library(e1071)
library(caret)
library(ggplot2)
set.seed(42)
n <- 100
data <- data.frame(
 x1 = rnorm(n),
 x2 = rnorm(n),
 class = factor(sample(0:2, n, replace = TRUE))
)
trainIndex <- createDataPartition(data$class, p = 0.7, list = FALSE)
train_data <- data[trainIndex, ]
test_data <- data[-trainIndex, ]
cat("KSMSCIT033 - Ashish Shende\n")
cat("========================\n")
head(data,5)
```

```
KSMSCIT033 - Ashish Shende
========================
```

A data.frame: 5 × 3

	x1	x2	class
	<dbl>	<dbl>	<fct>
1	1.3709584	1.2009654	0
2	-0.5646982	1.0447511	1
3	0.3631284	-1.0032086	0
4	0.6328626	1.8484819	2
5	0.4042683	-0.6667734	1

```r
svm_model <- svm(class ~ x1 + x2, data = train_data, kernel = "linear")
cat("KSMSCIT033 - Ashish Shende\n")
cat("========================\n")
predictions <- predict(svm_model, test_data)
accuracy <- mean(predictions == test_data$class)
cat(sprintf("Model accuracy on test set: %.2f%%\n", accuracy * 100))
w <- t(svm_model$coefs) %*% svm_model$SV
b <- svm_model$rho
cat(w,b)
```

```
KSMSCIT033 - Ashish Shende
========================
Model accuracy on test set: 32.14%
0.4989459 1.028644 3.358455 -1.924063 0.6553173 -0.1622308 -0.7810305
```

```r
cat("KSMSCIT033 - Ashish Shende\n")
cat("========================\n")
ggplot(data, aes(x = x1, y = x2, color = class)) +
 geom_point(size = 3, shape = 16) + # Plot data points
 geom_point(data = svm_model$SV, aes(x = x1, y = x2), shape = 17, size = 4, color = "black") + # Support Vectors
 geom_abline(slope = -w[1] / w[2], intercept = -b / w[2], color = "black", linewidth = 1) +
 scale_color_manual(values = c("red", "blue", "green")) + # Ensure 3 distinct colors for 3 classes
 theme_minimal() +
 labs(title = "SVM Classifier with Decision Boundary and Support Vectors", color = "Class")
```

```
KSMSCIT033 - Ashish Shende
========================
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

Practical 6A: Implementation of K-Means in Python

```
#6A
print("KSMSCIT033 - Ashish Shende")
print("========================")
x=[4,5,10,4,3,11,14,6,10,12]
y=[21,19,24,17,16,25,24,22,21,21]
print('\033[1mUnlabelled data\033[0m')
print('x = ',x)
print('y =' ,y)
```

```
KSMSCIT033 - Ashish Shende
========================
Unlabelled data
x =  [4, 5, 10, 4, 3, 11, 14, 6, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
```

```
import matplotlib.pyplot as plt
plt.scatter(x,y)
print("KSMSCIT033 - Ashish ShendeL")
print("========================")
plt.show()
```

```
KSMSCIT033 - Ashish ShendeL
========================
```

```
from sklearn.cluster import KMeans
data = list(zip(x,y))
print("KSMSCIT033 - Ashish Shende")
print("========================")
print('\033[1mCordinates\033[0m')
print(data)
```

```
KSMSCIT033 - Ashish Shende
========================
Cordinates
[(4, 21), (5, 19), (10, 24), (4, 17), (3, 16), (11, 25), (14, 24), (6, 22), (10, 21), (12, 21)]
```

```
from sklearn.cluster import KMeans
kmeans = KMeans (n_clusters=2)
kmeans.fit(data)
plt.scatter(x, y, c=kmeans.labels_)
plt.title("K-Means with 2 Clusters")
print("KSMSCIT033 - Ashish Shende")
print("========================")
plt.show()
```

```
KSMSCIT033 - Ashish Shende
========================
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

Practical 6B: Implementation of K-Means in R

```
cat("KSMSCIT033 - Ashish Shende\n")
cat("========================\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)$tot.withinss)}
plot(1:10, wcss, type = "b", pch = 19, col = "blue",
 xlab = "Number of Clusters", ylab = "WCSS",
 main = "Elbow Method for Finding Optimal K: KSMSCIT033 - Ashish Shende")
set.seed(123)
kmeans_result <- kmeans(df, centers = 3, nstart = 25)
pca_result <- prcomp(df, scale. = TRUE)
df_pca <- data.frame(pca_result$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), KSMSCIT033 - Ashish Shende")
```

```
KSMSCIT033 - Ashish Shende
========================
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

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

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
Weka GUI Ch
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

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

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.

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.