Lab 0

5/3/25

TO 00

① Initialise values directly into dataframe

```
import pandas as pd
data = {
    'USN': ['1BM22CS100', '1BM22CS200',
            '1BM22CS300', '1BM22CS400',
            '1BM22CS500'],
    'Name': ['Alice', 'Bob', 'Charlie', 'David',
             'Eve'],
    'Marks': [85, 78, 92, 88, 76] }
df = pd.DataFrame(data)
print(df)
```

O/P

	USN	Name	Marks
0	1BM22CS100	Alice	85
1	1BM22CS200	Bob	78
2	1BM22CS300	Charlie	92
3	1BM22CS400	David	88
4	1BM22CS500	Eve	76

② Importing datasets from sklearn.datasets

```
from sklearn.datasets import load_diabetes
dia = load_diabetes()
df = pd.DataFrame(dia.data, column = dia.feature_names)
df.head()
```

O/P

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
0	0.03	0.05	0.06	0.02	-0.04	-0.4	-0.0	-0.01	-0.017	0.019

③ Importing datasets from a specific .csv file

```
file = 'sample - sales - data . csv'
df = pd. read - csv ( file)
df. head ( )
```

O/P

	Product	Quantity	Price	Sales	Region
0	laptop	5	1000	5000	North
1	mouse	15	20	300	West

④ Download dataset from repositories

```
file = " Dataset of diabetes . csv'
df = pd . read - csv ( file)
df . head ( )
```

O/P

	ID	No_Patient	Gender	AGE	Urea	Cr	HbA1c	Chol	TG
0	J02	17975	F	50	4.7	46	4.9	4.2	0.9

HDL	LDL	VLDL	BMI	CLASS
2.9	1.4	0.5	24.0	N

To Do : Stock Market Data Analysis

```
import yfinance as yf
import pandas as pd
import matplotlib . pyplot as plt
tickers = [ 'HDFCBANK . NS', 'ICICIBANK . NS',
            'KOTAKBANK . NS' ]
data = yf . download ( tickers , start = "2024 - 01 - 01",
       end = "2024 - 12 - 31", group_by = 'tickers' )
data . head ( )
print ( data . shape )
print ( data . columns )
```

```
hdfc_data = data['HDFCBANK.NS']
print(hdfc_data.describe())
hdfc_data['Daily Return'] = hdfc_data['close'].pct_change()
plt.figure(figsize=(12,8))
plt.subplot(2,1,1)
hdfc_data['close'].plot(title="HDFC - Closing Price",
                        color='blue')
plt.ylabel('Closing Price')
plt.subplot(2,1,2)
hdfc_data['Daily Return'].plot(title="HDFC -
                               Daily Returns", color="red")
plt.ylabel('Daily Returns')
plt.tight_layout()
plt.show()
```

Repeat for ICICI and KOTAK Banks

Lab 1

5/3/25

To DO

```
import pandas as pd
data = pd.read_csv('housing.csv')
print(data.info())
print(data.describe())
print(data['Ocean Proximity'].value_counts())
print(data.isnull().sum()[data.isnull().sum()>0])
```

O/P ocean_proximity

ISLAND	9136
INLAND	6551
NEAR OCEAN	2658
NEAR BAY	2290

O/P total_bedrooms 207

Sen 05.03.25

TODO - Diabetes Dataset

① df = pd.read_csv('diabetes.csv')

② Missing values :

val = df.isnull().sum()

O/P : Series ([], dtype : int64)

③ Handling missing values

num_col = df.select_dtypes(include = ['float64', 'int64'])
.columns

imputes = SimpleImputer(strategy = 'mean')

df[num_col] = imputes.fit_transform(df[num_col])

cat_col = df.select_dtypes(include = ['object']).
columns

imputes_cat = SimpleImputer(strategy = 'most_frequent')

df[cat_col] = imputes_cat.fit_transform(df[cat_col])

④ Handling Categorical Attributes

df1 = df.copy()

df1['Gender'] = df1['Gender'].str.upper()

ordinal_encoder = OrdinalEncoder(categories = [['M', 'F']])

df1['Gender_encoded'] = ordinal_encoder.fit_transform
(df1[['Gender']])

onehot_encoder = OneHotEncoder()

encoded_data = onehot_encoder.fit_transform
(df1[['CLASS']])

encoded_array = encoded_data.toarray()

encoded_df = pd.DataFrame(encoded_array,
columns = onehot_encoder.
get_feature_names_out
(['CLASS']))

df_encoded = pd.concat([df1, encoded_df], axis=1)

df_encoded.drop(['cardio', 'class'], axis=1, inplace=True)

⑤ Data Normalization

norm = MinMaxScaler()

df_encoded[['Urea']] = normalizer.fit_transform(df_encoded[['Urea']])

scaler = StandardScaler()

df_encoded[['AGE']] = scaler.fit_transform(df_encoded[['AGE']])

⑥ Removing Outliers

df1 = df_encoded

Q1 = df1['Urea'].quantile(0.25)

Q3 = df1['Urea'].quantile(0.75)

IQR = Q3 - Q1

lower = Q1 - 1.5 * IQR

upper = Q3 + 1.5 * IQR

df1['Urea'] = np.where(df1['Urea'] > upper, upper, np.where(df1['Urea'] < lower, lower, df1['Urea']))

df2 = df_encoded

df2['Urea_zscore'] = stats.zscore(df2['Urea'])

df2['Urea'] = np.where(df2['Urea_zscore'].abs() > 3, np.nan, df2['Urea'])

```
df3 = df_encoded
df3['Area_zscore'] = stats.zscore(df3['Area'])
median_area = df3['Area'].median()
df3['Area'] = np.where(df3['Area_zscore'].abs()
                       > 3, median_area, df3['Area'])
```

12/3/25

Lab 2

(5)

```
import pandas as pd
from collections import Counter

def entropy(data):
    labels = data['label'].tolist()
    counts = Counter(labels)
    probabilities = [count / len(labels) for count in
                     counts.values()]
```

(6)

```
    entropy_value = -sum(p * math.log2(p) for
                    p in probabilities if p > 0)
    return entropy_value

def gain(data, feature):
    initial_entropy = entropy(data)
    feature_values = data[feature].unique()
    weighted_entropy = 0
    for value in feature_values:
        subset = data[data[feature] == value]
        weighted_entropy += (len(subset) / len(data))
                            * entropy(subset)
    return initial_entropy - weighted_entropy
```

```
def id3(data, features, target_attribute):
    if len(data['label'].unique()) == 1
        return data['label'].iloc[0]

    if len(features) == 0:
        return data['label'].value_counts().index[0]

    best_feature = max(features, key=lambda feature:
                       gain(data, feature))

    tree = {best_feature: {}}

    features = [f for f in features if f != best_feature]

    for value in data[best_feature].unique():
        subset = data[data[best_feature] == value]
                 .drop(columns=[best_feature])

        if len(subset) == 0:
            tree[best_feature][value] = data['label'].
                                        value_counts().index[0]
        else:
            tree[best_feature][value] = id3(subset,
                                        features, target_attribute)

    return tree
```

```
import math

data = {}

df = pd.DataFrame(data)

features = ['outlook', 'temperature', 'humidity', 'wind']

target_attribute = 'label'

decision_tree = id3(df, features, target_attribute)
```

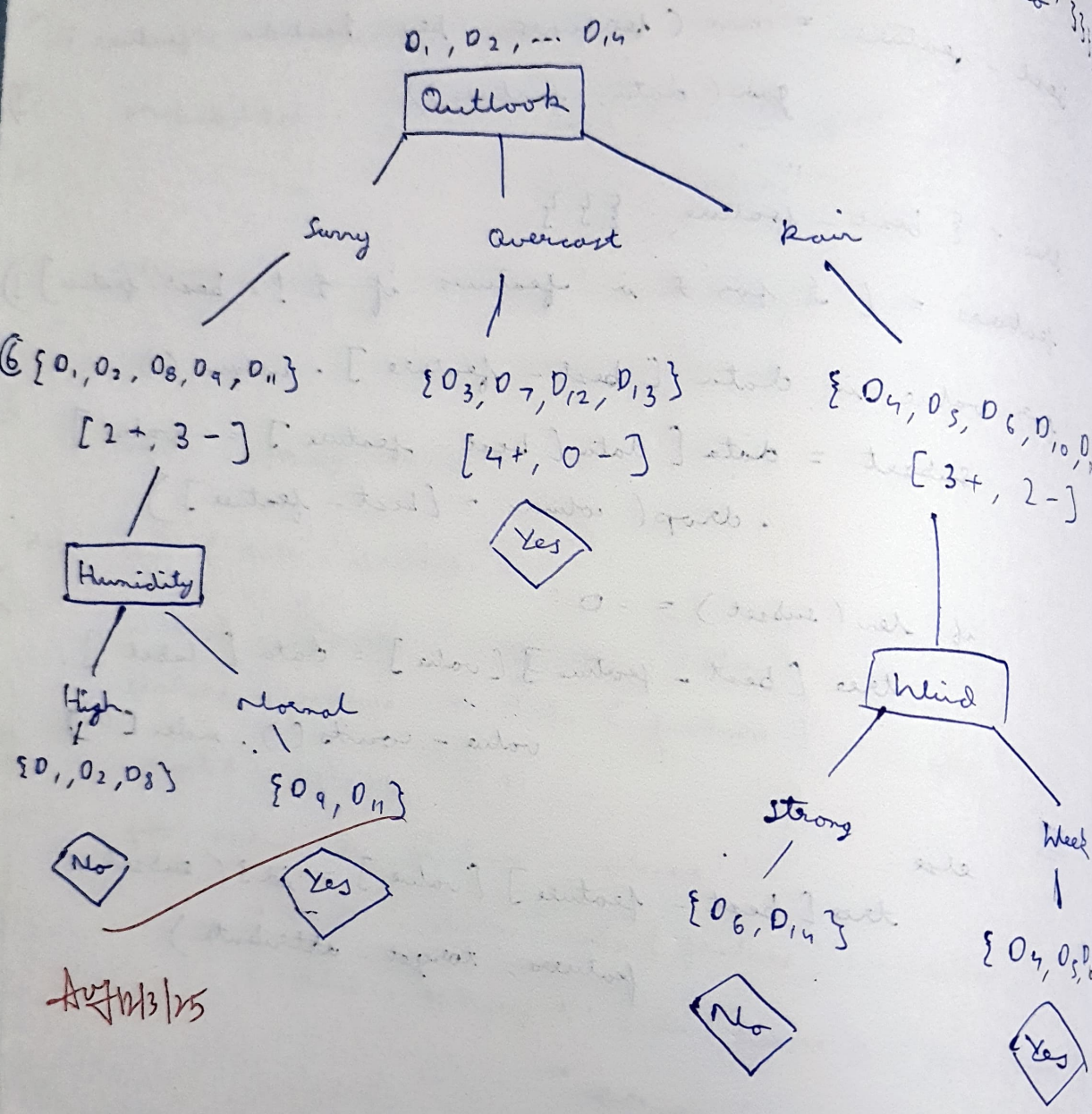
Q1

{'outlook': {'sunny': {'humidity': {'high': 'no', 'normal': 'yes'}},

'overcast': 'yes',

'rainy': {'wind': {'weak': 'yes', 'strong': 'no'}}}}

⑤



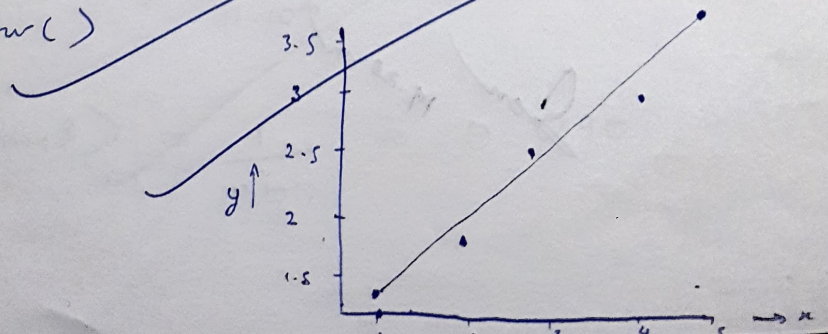
⑥

Answered 12/3/25

Lab 3

Linear Regression

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([1,2,3,4,5])
y = np.array([1.2, 1.8, 2.6, 3.2, 3.8])

x_b = np.c_[np.ones((x.shape[0], 1)), x]

theta = np.linalg.inv(x_b.T.dot(x_b)).dot(x_b.T)
        .dot(y)

intercept, slope = theta

x_input = float(input("Enter a value for x to
                      predict y: "))

x_input_b = np.array([1, x_input])

y_pred = x_input_b.dot(theta)

plt.scatter(x, y, color='blue', label='Data points')
plt.plot(x, x_b.dot(theta), color='red',
         label='Regression Line')

plt.xlabel('x')
plt.ylabel('y')
plt.title('Linear Regression - Model fit')
plt.legend()
plt.show()
```



Output :

Intercept (theta) : 0.54 0000.025

⑤ Slope (theta 1) : 0.6600 -- 01

Enter a value for x to predict y : 7

Predicted y value for x = 7.0 : 5.16000...02

Linear regression - Gradient Descent

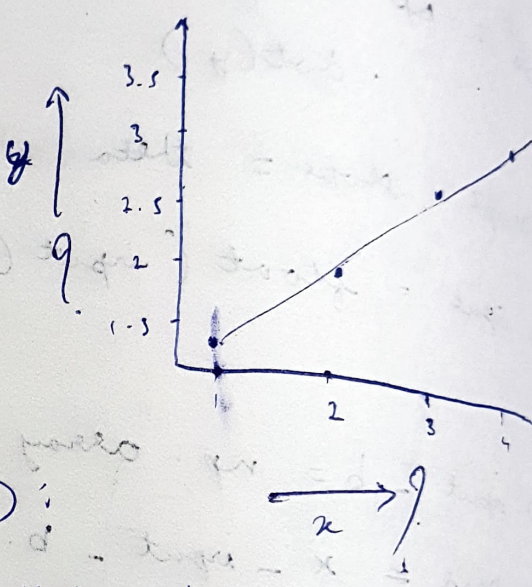
alpha = 0.01

iterations = 1000

m = len(x)

theta 0 = 0

theta 1 = 0



for _ in range (iterations):

y_pred = theta 0 + theta 1 * x

error = y_pred - y

theta 0 -= alpha * (1/m) * np.sum(error)

theta 1 -= alpha * (1/m) * np.sum(error * x)

x_input = float(input())

y_pred = theta 0 + theta 1 * x_input

19.03

02-04-25

Lab 4 - Logistic Regression

① Solution:

a) $a_0 = -5$

$a_1 = 0.8$

$$P(\text{pass}) = \frac{1}{1 + e^{-(-5 + 0.8x)}}$$

b) $a_0 + a_1 \cdot 7 = -5 + 0.8 \cdot 7 = 0.6$

$$P(\text{pass}) = 0.645$$

c) $P(\text{pass}) > 0.5 \rightarrow$ Hence "pass"

② $$P(\text{class } i) = \frac{e^{z_i}}{\sum_j e^{z_j}}, \quad z = [2, 1, 0]$$

$$\sum_j e^{z_j} = e^2 + e^1 + e^0 \approx 11.107$$

$e^{z_1} = e^2 = 7.389$

$e^{z_2} = e^1 = 2.718$

$e^{z_3} = e^0 = 1$

$$P(\text{class } 1) = \frac{e^2}{11.107} \approx 0.666$$

$$P(\text{class } 2) = \frac{e^1}{11.107} \approx 0.245$$

$$P(\text{class } 3) = \frac{1}{11.107} \approx 0.090$$

1. (i) Satisfaction level has highest import due to highest correlation factor.

(ii) It is 77.07%. which is not good because it is not a proper correlation

2. (i) Yes, to get all numeric data

(ii) No, but I dropped it from table if it was present

(iii) It is very accurate, with ~~about~~ no false predictions

(iv) Class label 4 due to inconsistent data & pattern

Lab - 6 - KNN

02-04-25

Person	Age	Salary	Target	Distance	Rank
A	18	50	N	52.8	5
B	23	55	N	46.6	4
C	24	70	N	31.9	2
D	41	60	Y	40.5	3
E	43	70	Y	31.1	1
F	38	40	Y	60.1	6

$k = 3$, Closest 3 : C, E, D

2 Y's and 1 N → ∴ predicted value for $X = (35, 100) = Y$

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$ where

$(x_1, y_1) = (35, 100)$

① Iris :

For $k = 5$, accuracy = 0.9667
$k = 3$ error = 0.0333

For $k = 4$, accuracy = 1.000
error = 0.000

For $k = 6$, accuracy = 0.9333
error = 0.0667

② Diabetes

It is important as it is distance based.

Otherwise cannot be scaled properly.

Standard scaler : $z = \frac{u - x}{s}$

u = mean

s = std.

x = feature value

16/04/25

Lab - 7 : Random Forest

S1: Build a Random Forest

a) If no of examples in the train set is N, take a sample of n-examples of random but with replacement, from original data. This sample will be training set for generating the tree

Again out of 100, choose next 30 examples, randomly for 2nd ip & put from back.

S2 : If there are M i/p vars., m vars. are selected at random, out of M, the best split on these 'm', used to split node. The value of 'm' held constant during the generation of various trees in forest

Ex: 10 i/p vars → 5
select subset of these vars. randomly to build tree

S3 : Each tree is grown to the largest extent

For new data, pass find prediction of each branch & assign new data part to category that with majority votes, i.e. wherever a new data pt. is given, give that DP to the particular DT

16.04

Lab 8 - Adaboost

7/5/205

1

(1) Assign each item = 1/6 weight

CGPA	Interactivities	Practical knowledge	Communication Skill	Job Profile
>=9	Yes	Good	Good	Yes
<9	No	Good	Moderate	Yes
>=9	No	Avg	Moderate	No
<9	No	Avg	Good	No
>=9	Yes	Good	Moderate	Yes
>=9	Yes	Good	Moderate	Yes

2

CGPA >=9 → Yes
<9 → No

CGPA	Predicted Job Offer	Actual Job Offer	Weight
>=9	Yes	Yes	1/6
<9	No	Yes	1/6
>=9	Yes	No	1/6
<9	No	No	1/6
>=9	Yes	Yes	1/6
>=9	Yes	Yes	1/6

$$\varepsilon_{CGPA} = 2 \times \frac{1}{6} = 0.333$$

$$\alpha_{CGPA} = \frac{1}{2} \frac{\ln(1-0.333)}{0.333} = 0.347$$

$$Z_{CGPA} = \frac{1}{6} \times 4 \times e^{-0.347} + \frac{1}{6} \times 2 \times e^{0.347}$$

$$= 0.9428$$

$$wt(d_j)_{i+1} = \frac{\frac{1}{6} \times e^{-0.347}}{0.9428} = 0.1239$$

$$wt(d_j)_{i+1} = \frac{\frac{1}{6} \times e^{0.349}}{0.9428} = 0.2501$$

Interactiviess	Interactiviess	Predicted Job offer	Actual Job Offer	Weight
Yes → Yes				
No → No	Yes	Yes	Yes	0.1249
	No	No	Yes	0.2501
	No	No	No	0.2501
	No	No	No	0.1249
	Yes	Yes	Yes	0.1249
	Yes	Yes	Yes	0.1249

$$\varepsilon_{Interact} = 1 \times 0.2501 = 0.2501$$

$$\alpha_{Interact} = \frac{1}{2} \frac{\ln(1-0.2501)}{0.2501} = 0.5490$$

$$Z_{interact} = 0.1249 \times 4 \times e^{-0.549} + 0.2501 \times 1 \times e^{-0.549} + 0.2501 \times 1 \times e^{0.549}$$

$$= 0.866$$

$$wt(d_j)_{i+1} = \frac{0.1249 \times e^{-0.549}}{0.866} = 0.0832$$

$$wt(d_j)_{i+1} = \frac{0.2501 \times e^{-0.549}}{0.866} = 0.1667$$

$$wt(d_j)_{i+1} = \frac{0.2501 \times e^{0.549}}{0.866} = 0.5001$$

Practical knowledge

Good → Yes

Moderate → No

Average

No instances misclassified, no need decision stump

Communication

Good → Yes

Moderate → No

Comm Skill	Predicted Job Offer	Actual Job Offer	Weight
Good	Yes	Yes	0.0832
Moderate	No	Yes	0.5001
Moderate	No	No	0.1667
Good	Yes	No	0.0832
Moderate	No	Yes	0.0832
Moderate	No	Yes	0.0832

$$\xi_{CS} = 1 * 0.5001 + 3 * 0.0832 = 0.7497$$

$$\alpha_{CS} = \frac{1}{2} \frac{\ln(1-0.7497)}{0.7497} = -0.5485$$

$$Z_{CS} = 0.0832 \times 1 \times e^{-(-0.5485)} + 0.1667 \times 1 \times e^{-(-0.5485)} + 0.5001 \times 1 \times e^{-0.5485} + 0.0832 \times 3 \times e^{(-0.5485)}$$

$$Z_{CS} = 0.866$$

$$wt(d_j)_{i+1} = \frac{0.0832 \times e^{-(-0.5485)}}{0.866} = 0.1663$$

$$wt(d_j)_{i+1} = \frac{0.1667 \times e^{-(-0.5485)}}{0.866} = 0.3331$$

$$wt(d_i)_{i+1} = \frac{0.5001 * e^{(-0.5485)}}{0.866} = 0.3337$$

$$wt(d_i)_{i+1} = \frac{0.0832 * e^{(-0.5485)}}{0.8666} = 0.0555$$

Final

$$H_F(d_i) = \alpha_{CGPA} * Yes + \alpha_{Internet} * Yes + \alpha_{CSE} * Yes$$

$$= 0.347 * 1 + 0.5490 * 1 - 0.5485 * 1 = 0.34$$

Sl No.	$\alpha_{CGPA} = 0.347$	$\alpha_{Internet} = 0.5490$	$\alpha_{Commskill} = -0.5485$	Weight Avg.	Final Prediction
1	Yes	Yes	Yes	0.3475	Yes
2	No	No	No	0.0	No
3	Yes	No	No	0.347	Yes
4	No	No	Yes	-0.5485	No
5	Yes	Yes	No	0.896	Yes
6	Yes	Yes	No	0.896	Yes

(2) Best accuracy = 0.83 or 83%.

Nr. of trees = 160.

Confusion matrix = $\begin{bmatrix} 10658 & 45.1 \\ 2023 & 1521 \end{bmatrix}$

7/5/25

Lab 9 - K mean Clustering

	A	B
Record		
R1	1.0	1.0
R2	1.5	2.0
R3	3.0	4.0
R4	5.0	7.0
R5	3.5	5.0
R6	4.5	5.0
R7	3.5	4.5

k = 2, centroid for C1 = (1, 1).

" " C2 = (5, 7)

Iteration 1

Record No.	Close to C1 (1,1)	Close to C2 (5,7)	Assign to
R1 (1, 1)	dist(R1, C1) = 0.0	dist(R1, C2) = 7.21	C1
R2 (1.5, 2)	dist(R2, C1) = 1.12	dist(R2, C2) = 6.12	C1
R3 (3, 4)	dist(R3, C1) = 3.61	dist(R3, C2) = 3.61	C1
R4 (5, 7)	dist(R4, C1) = 7.21	dist(R4, C2) = 0.0	C2
R5 (3.5, 5)	dist(R5, C1) = 4.12	dist(R5, C2) = 2.5	C2
R6 (4.5, 5)	dist(R6, C1) = 5.31	dist(R6, C2) = 2.06	C2
R7 (3.5, 4.5)	dist(R7, C1) = 4.30	dist(R7, C2) = 2.92	C2

$C1\{R1, R2, R3\} = \left(\frac{1+1.5+3}{3}, \frac{1+2+4}{3}\right)$

$= \left(\frac{5.5}{3}, \frac{7.0}{3}\right)$

$= (1.83, 2.33)$

$$c2 = \left(\frac{5 + 3.5 + 4.5 + 3.5}{4}, \frac{7 + 5 + 5 + 4.5}{4}\right)$$

$$= \left(\frac{16.5}{4}, \frac{21.5}{4}\right)$$

$$= (4.12, 5.37)$$

Iteration 2

Record No.	Close to C1 (1.83, 2.33)	Close of C2 (4.12, 5.37)	Assign to Clusters
R1 (1,1)	dist (R1, C1) = 1.57	dist (R1, C2) = 5.37	C1
R2 (1.5, 2)	dist (R2, C1) = 0.47	dist (R2, C2) = 4.27	C1
R3 (3,4)	dist (R3, C1) = 2.04	dist (R3, C2) = 1.77	C2
R4 (5,7)	dist (R4, C1) = 5.64	dist (R4, C2) = 1.85	C2
R5 (3.5,5)	dist (R5, C1) = 3.15	dist (R5, C2) = 0.72	C2
R6 (4.5,5)	dist (R6, C1) = 3.78	dist (R6, C2) = 0.53	C2
R7 (3.5,4.5)	dist (R7, C1) = 2.74	dist (R7, C2) = 1.07	C2

$$C1 = \left(\frac{1 + 1.5}{2}, \frac{1 + 2}{2}\right)$$

$$= 1.25, 1.5$$

$$C2 = \left(\frac{3 + 5 + 3.5 + 4.5 + 3.5}{5}, \frac{4 + 7 + 5 + 5 + 4.5}{5}\right)$$

$$= (3.9, 5.1)$$

$C_1 \{R_1, R_2\}$

$C_2 \{R_3, R_4, R_5, R_6, R_7\}$

Lab 10 - PCA

heart.csv

Report accuracy :

	Before PCA	After PCA
SVM Acc.	0.1648	0.1848
Logistic Regression Acc	0.1685	0.1739
Random Forest	0.1848	0.1738

08.06.2024