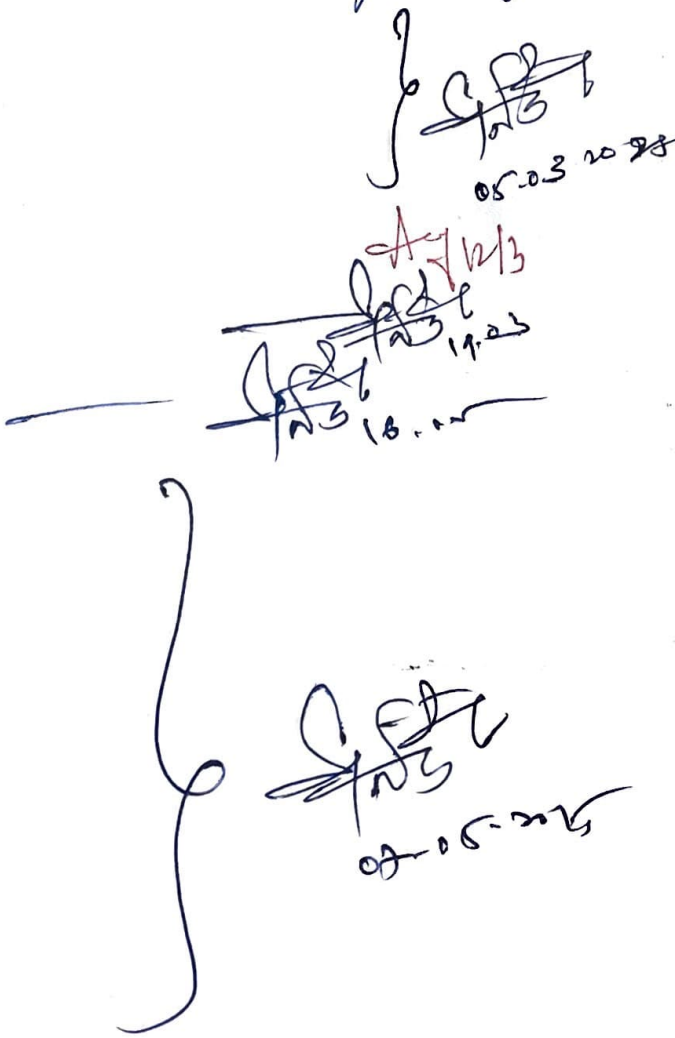
S-No.	Date	Title	~~Page No~~ Sign
1.	5/03/25	Lab 0	
2.	5/03/25	Lab 1	
3.	12/03/25	Lab 2	
4.	19/03/25	Lab 3	
5.	16/04/25	Lab 4	
6.	30/04/25	Lab 5	
7.	30/04/25	Lab 7	
8.	7/05/25	Lab 8	
9.	7/05/25	Lab 9	
10.	7/05/25	Lab 10	



5/3/25 Lab 0

```
from google.colab import drive
drive.mount('/content/drive')
```

To do

Method 1 : Initializing values directly into DataFrame

```
import pandas as pd
data = {
    'USN': ['1BS17CS001', '1BS17CS002', '1BS17CS003'],
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Marks': [85, 78, 92] }
df = pd.DataFrame(data)
df
```

Output :

	USN	Name	Marks
0	1BS17CS001	Alice	85
1	1BS17CS002	Bob	78
2	1BS17CS003	Charlie	92

Method 2 :

```
import pandas as pd
from sklearn.datasets import load_diabetes

diabetes_data = load_diabetes()

df = pd.DataFrame(diabetes_data.data, columns =
    diabetes_data.feature_names)

df['target'] = diabetes_data.target

df.head()
```

Output : age sex bmi bp s1

bmi	bp	s1	s2	s3	s4	s5	s6	target
0.061696	0.021872	-0.044	-0.034821	-0.043401	-0.00259	0.019907	-0.017646	151.0

Method 3

```
import pandas as pd
file_path = '/content/drive/My Drive/Lab-0/sample_sales_data.csv'
df = pd.read_csv(file_path)
df.head()
```

Output:

	Product	Quantity	Price	Sales	Region
0	Laptop	5	1000	5000	North
1	Mouse	15	20	300	West

Method 4

```
#STOCK MARKET DATA ANALYSIS
import yfinance as yf
import matplotlib.pyplot as plt

tickers = ["HDFCBANK.NS", "ICICIBANK.NS", "KOTAKBANK.NS"]

data = yf.download(tickers, start="2024-01-01",
        end="2024-12-30", group_by='ticker')

data.head()
```

Summary statistics for a specific stock (3 banks - HDFC, ICICI, KOTAK):

```
hdfc = data['HDFCBANK.NS']
print(hdfc.describe())
```

similarly for ICICI, KOTAK

```
icici = data['ICICIBANK.NS']
icici['Daily Return'] = icici['Close'].pct_change()
```

Plotting closing price and daily returns:

```
plt.figure(figsize=(12,6))
plt.subplot(2,1,1)
hdfc['Close'].plot(title="HDFC - Closing Price")
plt.subplot(2,1,2)
hdfc['Daily Return'].plot(title="HDFC - Daily Returns",
    color='orange')
plt.tight_layout()
plt.show()
```

Similarly, for Kotak and ICICI

5/03/25

Lab 1

```
import pandas as pd
# Load .csv file
df = pd.read_csv("housing (1).csv")
# display column information
print(df.info())
# display statistical information of all numerical columns
print(df.describe())
# display count of unique labels for "Ocean Proximity"
if "ocean_proximity" in df.columns:
    print(df["Ocean_proximity"].value_counts())
else:
    print("not found")
# display columns with count of missing values >0
missing_values = df.isnull().sum()
columns_with_missing = missing_values[missing_values > 0]
print(columns_with_missing)
```

06.03.2025

TO-DO - 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

imputer = SimpleImputer(strategy='mean')
df[num_col] = imputer.fit_transform(df[num_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(['Gender', 'Class'], axis=1, inplace=True)
```

(5) Data Normalisation

```
norm = MinMaxScaler()
df-encoded[('Urea')] = normaliser.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 = Q1 + 1.5*IQR
df1['Urea'] = np.where(df1['Urea'] > upper,
    upper, np.where(df1['urea'] < lower,
                    lower, df['urea']))

df2 = df_encoded
df2['urea-zscore'] = (df2['Urea'])
df2['Urea'] = np.where(df2['Urea-zscore'].
    abs() > 3, np.nan,
    df2['Urea'])
```

```
df3 = df_encoded
df3['urea_zscore'] = stats.zscore(df3['urea'])
median_urea = df3['urea'].median()
df3['urea'] = np.where(df3['urea_zscore'].abs()
                       > 3, median_urea, df3
                       ['urea'])
```

12/03/25

Lab 2

```
impat 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()]
    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
```

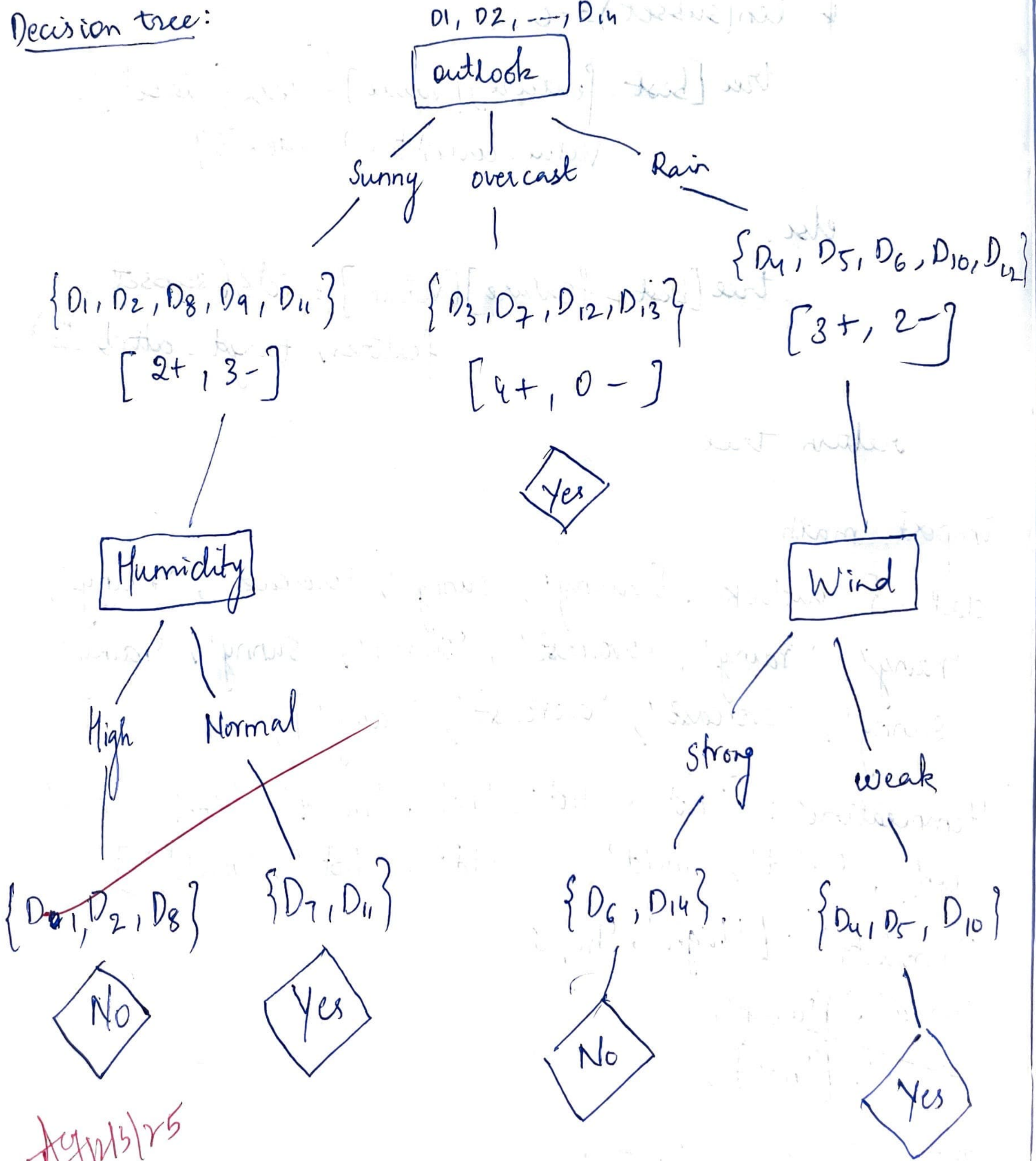
```
import math
data = {'outlook' : ['sunny', 'sunny', 'overcast', 'rainy',
 'rainy', 'rainy', 'overcast', 'sunny', 'sunny', 'rainy',
 'sunny', 'overcast', 'overcast', 'rainy'],
'temperature' : ['hot', 'hot', 'hot', 'mild', 'cool', 'cool',
 'cool', 'mild', 'mild', 'mild', 'hot', 'mild'],
'humidity' : ['high', 'high'
'wind' : ['weak',
'label' : ['no',                                    ]}
df = pd.DataFrame(data)
features = ['outlook', 'temperature', 'humidity', 'wind']
target_attribute = 'label'
```

decision_tree = id3 (df, features, target_attribute)

print (decision_tree)

Output: {'outlook': {'Sunny': {'humidity': {'high': 'no', 'normal': 'yes'}}, 'overcast': 'yes', 'rainy': {'wind': {'weak': 'yes', 'strong': 'no'}}}}

Decision tree:



Ag 12/3/25

19/03/25

Lab 3

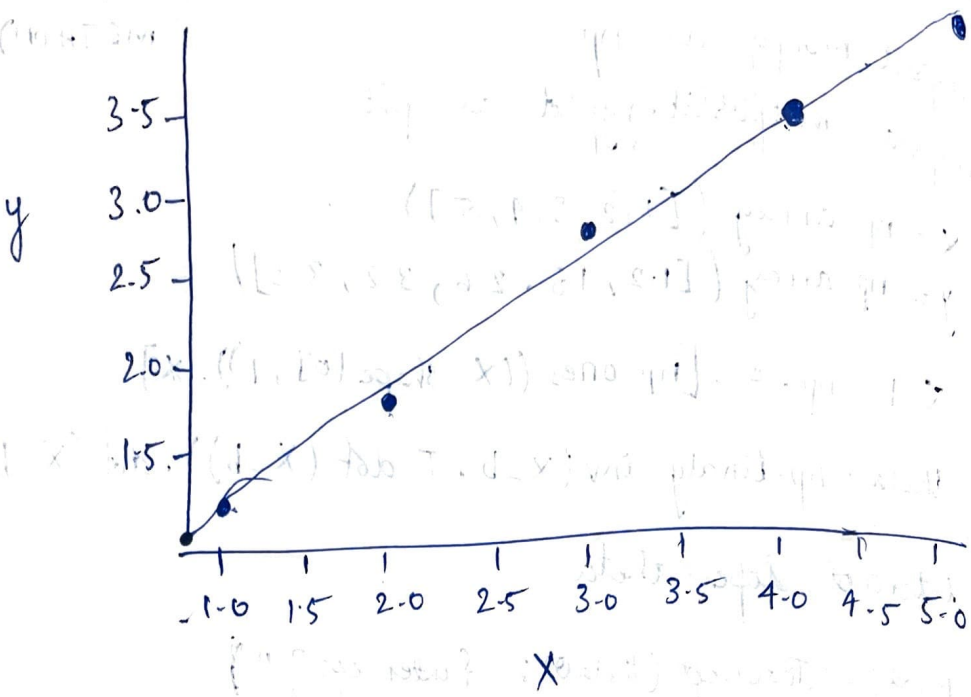
Linear Regression (MATRIX METHOD)

```
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
print(f"Intercept (theta0): {intercept}")
print(f"Slope (theta1): {slope}")
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)
print(f"Predicted y for X = {x_input}: {y_pred}")
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 (theta0) : 0.5400000000000002
Slope (theta1) : 0.66
Enter a value for X to predict y: 5
```

Predicted y for X = 5.0 : 3.840000...34



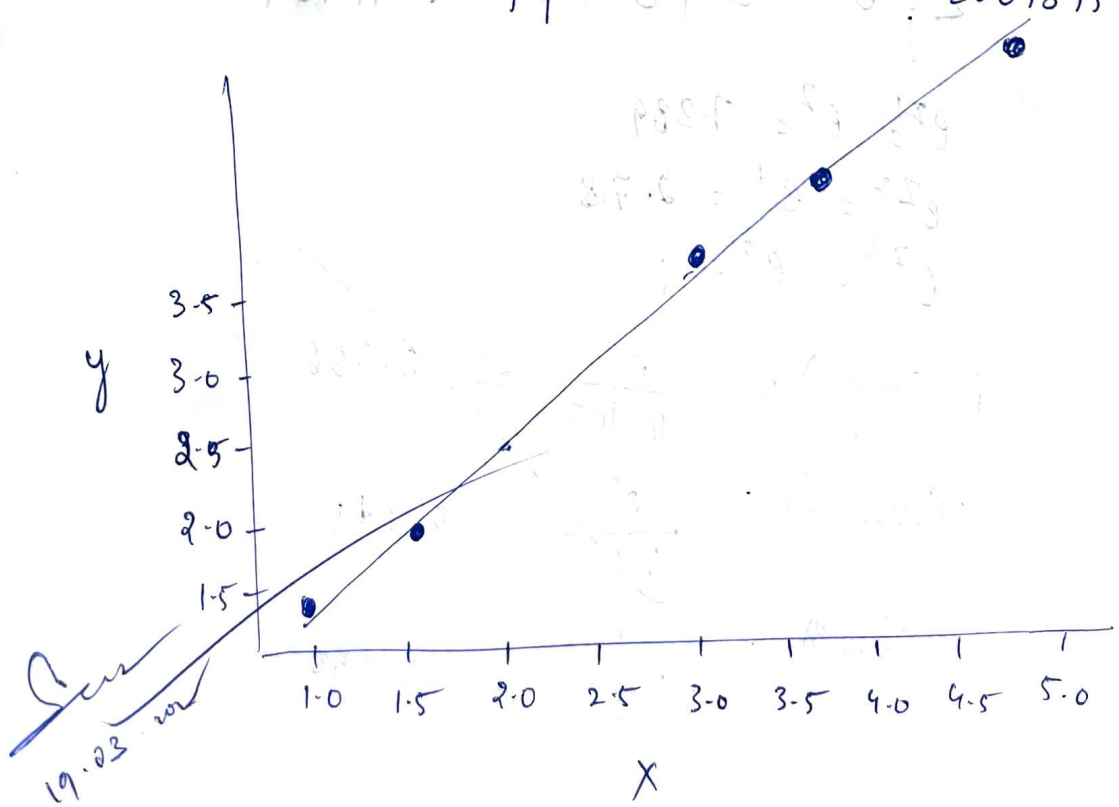
(2)

```
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])

alpha = 0.01
iterations = 1000
m = len(x)

theta0 = 0
theta1 = 0

for _ in range(iterations):
    y_pred = theta0 + theta1 * x
    error = y_pred - y

    theta0 -= alpha * (1/m) * np.sum(error)
    theta1 -= alpha * (1/m) * np.sum(error * x)
```

```
print(f"Intercept (theta 0): {theta0}")
print(f"Slope (theta 1): {theta1}")

x_input = float(input("Enter a value for X to predict y:"))
y_pred = theta0 + theta1 * x_input

print(f"Predicted y for X = {x_input}: {y_pred}")

plt.scatter(X, y, color='blue', label='Data points')
plt.plot(X, theta0 + theta1 * X, color='red',
         label='Regression line')
plt.xlabel('X')
plt.ylabel('y')
plt.title('Linear Regression')
plt.legend()          plt.show()
```

Output:

```
Intercept (theta 0) = 0.47890448451315504
Slope (theta 1): 0.6769224781201147
Enter a value for X to predict y: 9
Predicted y for X = 9.0: 6.571206787594183
```



19.03.2025

2/4/25

Lab 4 (Logistic Regression)

(1) $a_0 = -5$ (intercept) $a_1 = 0.8$ (slope)

a. $Z = \frac{1}{e^{-(-5+0.8x)}}$ $\Rightarrow$ P(Pass)

$x = 7$

b. $Z = \frac{1}{e^{-(-5+0.8\times7)}}$

$P(Pass) = 0.645$

$P = \begin{cases} \geq 0.5 & \text{pass} \\ \text{otherwise} & \text{fail} \end{cases}$

c. $P(pass) > 0.5 \rightarrow$ Hence, "Pass".

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

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

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

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

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

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

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

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

Logistic Regression

1. HR_comma_sep.csv

(i) 'Satisfaction_level' as it had most direct impact of 'left' output variable

(ii) 77.07% accuracy

2. Zoo dataset

(i) Yes. check for missing (null) values in the dataset & dropping rows with missing values.

(ii) Accuracy of the model is 95%.

(iii) 4th class is the only misclassified class due to probable inconsistency in data & misunderstandings in relationships of the data.

KNN

a)

Person	Age	Salary(k)	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
X	35	100			

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

① Iris

$k=5 \Rightarrow$ 96.67% accuracy

3.33% error rate

$k=4 \Rightarrow$ 100% accuracy

0% error rate

$k=3 \Rightarrow$ 93.33% accuracy

6.67% error rate

② Diabetes

It is important as it is distance based.

Otherwise cannot be scaled properly.

Standard scalar : $Z = \frac{\mu - x}{s}$

μ = mean

s = std

x = feature value

Random Forest: (16-04-2024)

Algo: S1: Build a Random Forest

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

S2: If there are M input variables, 'm' variables are selected at random, out of M, then best split on these 'm' - used to split the node. The value at 'm' is held constant during the generation of various trees in the forest.

S3: Each tree is grown to the largest ~~extract~~ extent possible.

For new data point find the prediction of each DTs & ~~design~~ assign new data point to the category that wins the majority votes.

i.e. whenever a new data pt. is given give that DT to that particular DT.

16.04.24

7/05/25

Lab 8

AdaBoost

① Initial weight assigned to each item = 1/6

first Decision stump classifier

CGPA: based on 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

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

Weight of each weak classifier:

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

$$\alpha_{CGPA} = 0.347$$

Normalizing factor:

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

$$Z_{CGPA} = 0.9428$$

Update weights of all data instances:

$$wt(d_j)_{i+1} = \frac{\frac{1}{6} \times e^{-0.347}}{0.9428} = 0.1249$$

$$wt(d_j)_{i+1} = \frac{\frac{1}{6} \times e^{0.347}}{0.9428} = 0.2501$$

~~$\varepsilon_{Interact} = 1 * 0.2501 = 0.2501$~~

Interact: Yes → Yes
No → No

	Interact	Predicted Job offer	Actual Job offer	Weight
	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

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

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

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

$$\alpha_{Interact} = 0.5490$$

$$Z_{Interact} = 0.1249 * 4 * e^{-0.549} + 0.2501 * 1 * e^{-0.549} + 0.2501 * 1 * e^{0.549}$$

$$Z_{Interact} = 0.866$$

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

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

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

Practical Knowledge : Good → Yes

Aveage → No

Pok	Predicted	Actual	Weight
G	Yes	Yes	0.0832
AG	Yes	Yes	0.5001
A	No	No	0.1667
A	NO	No	0.0832
G	Yes	Yes	0.0832
G	Yes	yes	0.0832

No instances are misclassified

Communication Skill Good → Yes

Moderate → No

Comm Skill	Predicted	Actual	Weight
G	Yes	Yes	0.0832
M	No	Yes	0.5001
M	No	No	0.1667
G	Yes	No	0.0832
M	No	Yes	0.0832
M	No	Yes	0.0832

$$\varepsilon_{cs} = 1 \times 0.5001 + 3 \times 0.0832 = 0.7497$$

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

$$Z_{cs} = 0.0832 \times 1 \times e^{-(-0.5485)} + 0.1667 \times 1 \times e^{-(-0.5485)} + 0.5001 \times 1 \times e^{(-0.5485)} + 0.0832 \times 3 \times e^{+(-0.5485)} = 0.866$$

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

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

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

$$wt(d_j)_{i+1} = \frac{0.0832 \times e^{+(-0.5485)}}{0.866} = 0.0555$$

$$H_F(d_j) = \alpha_{CGPA} \times 1 + 0.5490 \times 1 - 0.5485 \times 1$$

$$= 0.34$$

$$H_F(d_j) = 0.347 \times 0 + 0.5490 \times 0 - 0.5485 \times 0$$

$$= 0.$$

$\alpha_{CGPA}=0.347$	$\alpha_{intmad}=0.5490$	$\alpha_{cs}=-0.5485$	Weighted Avg	Final Predict
Yes	Yes	Yes	0.3475	Yes
No	No	No	0.0	No
Yes	No	No	0.347	Yes
No	No	Yes	-0.5485	No
Yes	Yes	No	0.896	Yes
Yes	Yes	No	0.896	Yes

(2) Best accuracy = 0.83 ~ 83%.

No. of trees = 160

Confusion matrix = $\begin{bmatrix} 10658 & 457 \\ 2023 & 1521 \end{bmatrix}$

7/05/25

Lab 9

K Means

(1) Iteration 1

Record Number	Close to C1 (1.0, 1.0)	Close to C2 (5.0, 7.0)	Assign to Cluster
R1 (1.0, 1.0)	dist (R1, C1) = 0.0	(R1, C2) = 7.21	Cluster 1
R2 (1.5, 2.0)	dist (R2, C1) = 1.12	(R2, C2) = 6.12	Cluster 1
R3 (3.0, 4.0)	(R3, C1) = 3.61	(R3, C2) = 3.61	Cluster 1
R4 (5.0, 7.0)	(R4, C1) = 7.21	(R4, C2) = 0.0	Cluster 2
R5 (3.5, 5.0)	(R5, C1) = 4.21	(R5, C2) = 2.5	Cluster 2
R6 (4.5, 5.0)	(R6, C1) = 5.31	(R6, C2) = 2.06	Cluster 2
R7 (3.5, 4.5)	(R7, C1) = 4.3	(R7, C2) = 2.92	Cluster 2

Cluster 1 {R1, R2, R3}

Cluster 2 {R4, R5, R6, R7}

new centroids are:

C1 = (1.0+1.5+3.0)/3 , (1.0+2.0+4.0)/3

= 5.5/3 , 7.0/3

= 1.83 , 2.33

C2 = (5.0+3.5+4.5+3.5)/4 , (7+5+4.5)/4

= 16.5/4 , 21.5/4

= 4.12 , 5.37

Iteration 2:

Record No.	Close to C1 (1.83, 2.33)	Close to C2 (4.12, 5.37)	Assign to Cluster
R1(1.0, 1.0)	(R1, C1) = 1.57	(R1, C2) = 5.37	C1
R2(1.5, 2.0)	(R2, C1) = 0.47	(R2, C2) = 4.27	C1
R3(3.0, 4.0)	(R3, C1) = 2.04	(R3, C2) = 1.77	C2
R4(5.0, 4.0)	(R4, C1) = 5.64	(R4, C2) = 1.85	C2
R5(3.5, 5.0)	(R5, C1) = 3.15	(R5, C2) = 0.72	C2
R6(4.5, 5.0)	(R6, C1) = 3.78	(R6, C2) = 0.53	C2
R7(3.5, 4.5)	(R7, C1) = 2.74	(R7, C2) = 1.07	C2

Cluster 1: (R1, R2) & Cluster 2 (R3, R4, R5, R6, R7)

new centroids are:

C1 = (1.0 + 1.5)/2, (1.0 + 2.0)/2
= 2.5/2, 3.0/2
= 1.25, 1.5

C2 = (3.0 + 5.0 + 3.5 + 4.5 + 3.5)/5, (4 + 7 + 5 + 4.5)/5
= 19.5/5, 25.5/5
= 3.9, 5.1

7/05/25

Lab 10

PCA

heart.csv

	Before PCA	After PCA
SVM Accuracy	0.1848	0.1848
Logistic Regression Accuracy	0.1685	0.1739
Random Forest	0.1848	0.1739

Sri 07.05.2025