INDEX

NAME: SANJANA SURESH STD.: _____ SEC.: _____ ROLL NO.: _____ SUB.: ML LAB

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
01	05.03.25	WEEK - 0		
02	05.03.25	WEEK - 1		
03	12.03.25	LAB-02		
04	19.03.25	LAB - 03 Linear Regression		
05	02.04.2025	LAB - Logistic Regression		
06	02.04.2025	LAB - KNN Classification		
07	16.04.25	LAB - 7 Random Forest Classifier		
08	07.05.25	LAB - 08 AdaBoost		
09	07.05.25	LAB - 09 KMeans clustering		
10	07.05.25	LAB - 10 PCA		

05/03

WEEK - 00

TO-DO EXERCISES

① Initializing values directly into DataFrame - five rows of data with column heading as USN, Name, Marks.

```
import pandas as pd
data = { "USN" : ["234", "235", "236", "237", "238"],
         "Name" : ["Asha", "Cathy", "Rosa", "Amy", "Gina"],
         "Marks" : [85, 90, 78, 92, 88]
       }
df = pd.DataFrame(data)
print(df)
```

o/p

	USN	Name	Marks
0	234	Asha	85
1	235	Cathy	90
2	236	Rosa	78
3	237	Amy	92
4	238	Gina	88

② Importing datasets from sklearn.datasets

```
from sklearn.datasets import load_diabetes
import pandas as pd
diabetes = load_diabetes()
df_diabetes = pd.DataFrame(diabetes.data, columns = diabetes.feature_names)
print(df_diabetes)
```

o/p

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
0	0.03807	0.50680	0.061691	0.0218	-0.044	-0.034	-0.043		-0.01764	
1	-0.001882	-0.04472	-0.051474		-0.05					
2	0.085299	0.050680								
3										
4										

③ Import datasets from a specific .csv file

```
import pandas as pd
df_csv = pd.read_csv("/sample_sales_data.csv")
print(df_csv.head())
```

o/p

	Product	Quantity	Price	Sales	Region
0	Laptop	5	1000	5000	North
1	Mouse	15	20	300	West
2	Keyboard	10	50	500	East
3	Monitor	8	200	1600	South
4	Laptop	12	950	11400	North

④ Downloading datasets from existing dataset repository like Kaggle..

```
import pandas as pd
df_kaggle = pd.read_csv("/Dataset of Diabetes.csv")
print(df_kaggle.head())
```

o/p

	ID	No_Patient	Gender	AGE	Urea	Cr
0	502	17975	F	50	4.7	46
1	735	34221	M	26	4.5	62
2	420	47975	F	50	4.7	

	BMI	CLASS
0	24.0	N
1	23.0	N
2	24.0	N
3	24.0	N
4	21.0	N

TO - DO

Stock Market Data Analysis

1. ["HDFCBANK.NS", "ICICIBANK.NS", "KOTAKBANK.NS"]
2. start date and end date.
3. Plot the closing price and daily return for all three bank.

```
import yfinance as yf
import panda as pd
import pmatplotlib.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')
print ("First 5 rows of the dataset")
print ( data.head (1)
print ("\n Shape of dataset")
print (data.shape)
print ("\n Column names")
print ( data.columns)
hdfc_data = data ['HDFCBANK.NS']
print ("\n Summary statistics for HDFC Industries")
print (hdfc_data.describe ())
hdfc_data ['Daily Return'] = hdfc_data ['Close'].pct_change ()
plt.figure ( figsize = (12, 6))
plt.subplot (2, 1, 1)
hdfc_data ['Daily_Return'] = hdfc_data.plot (title = "HDFC Industries - Daily
  Returns", color - orang)
plt.tight_layout ()
plt.show ()
```

WEEK 1

① To load .csv file into dataframe

```
import pandas as pd
df = pd.read_csv('/content/housing.csv')
print(df.head())
print(df.columns)
print(df.info)
```

② To display information of all columns

```
print(df.columns())
```

	population	households	median_income	median_house_value	ocean_proximity
0	322.0	126.0	322.0		
1	2401.0	1138.0	8.304		
2	496.0		8.754		
3	558.0	219.0	5.6471		
4	565.0	219.0	3.8462		

③ To display statistical information of all numerical

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

④ To display the count of unique label for "Ocean Proximity" column

```
print(df.counter('ocean_proximity'))
```

⑤ To display which attributes in a dataset have missing values.

```
print(df.isnull())
```

	ocean_proximity
0	NEAR BAY
1	NEAR BAY
2	INLAND
⋮	

05-03-24

For Diabetes dataset, apply data preprocessing technique —

```
import pandas as pd
import numpy as np
import os
import sklearn.preprocessing import LabelEncoder, MinMaxScaler, StandardScaler

diabetes_df = pd.read_csv("/content/Dataset_of_Diabetes.csv")

# Check for missing value
print("Missing values in Diabetes dataset :\n", diabetes_df.isnull().sum())

# Handle missing values with mean
numeric_cols = diabetes_df.select_dtypes(include=np.number).columns
diabetes_df[numeric_cols] = diabetes_df[numeric_cols].fillna(diabetes_df[numeric_cols].
                                                             mean())

# To fill missing categorical values with mode
for col in diabetes_df.select_dtypes(include=['object']).columns:
    diabetes_df[col].fillna(diabetes_df[col].mode()[0], inplace=True)

# Identify categorical values and encode them.
categorical_cols = ['Gender', 'CLASS']
label_enc = LabelEncoder()
for col in categorical_cols:
    diabetes_df[col] = label_enc.fit_transform(diabetes_df[col])

# Apply min max scaling
min_max_scaler = MinMaxScaler()
diabetes_df_scaled = pd.DataFrame(min_max_scaler.fit_transform(diabetes_df),
                                  columns=diabetes_df.columns)
```

Apply standardization

std_scaler = StandardScaler()

① diabetes_df_standardized = pd.DataFrame(std_scaler.fit_transform(diabetes_df), columns = diabetes_df.columns)

Save the preprocessed datasets

diabetes_df_scaled.to_csv("/content/diabetes_preprocessed.csv", index = False)

os.makedirs("content", exist_ok = True)

diabetes_df_standardized.to_csv("/content/diabetes_standardized.csv", index = False)

print("Diabetes dataset preprocessing completed. Preprocessed datasets saved.")

②

Q1. Which columns in the dataset had missing values? How did you handle them?

- Numeric columns such as glucose levels, blood pressure, insulin.
- Categorical columns such as Gender and CLASS

To handle numeric data, mean of the respective column was used.

categorical data, mode

Q2. Which categorical columns did you identify in the dataset? How did you encode?

Gender and CLASS are the two categorical columns.

③ Encoding method - Label encoding Male → 0, Female → 1

Q3. What is the difference between min-max scaling and standardization?

④ • Min Max Scaling - transforms data to fixed range [0,1]

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

⑤ • Standardization - transforms data so that it has a mean of 0 and standard deviation of 1

$$x' = \frac{x - \mu}{\sigma}$$

19/03

LAB - 03

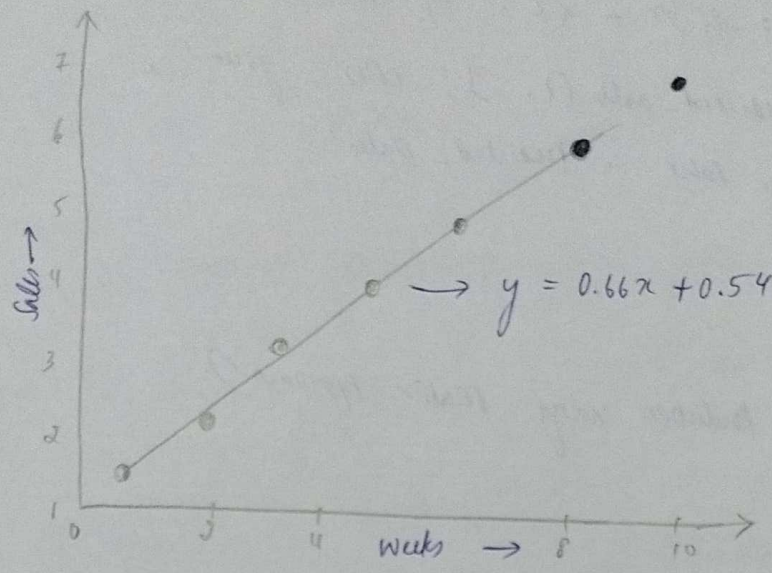
Build a linear regression model using 1) Simple Linear Regression
2) Linear regression in matrix

1) SIMPLE LINEAR REGRESSION

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

xi = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
yi = np.array([1.2, 1.8, 2.6, 3.2, 3.8])

model = LinearRegression()
model.fit(xi, yi)

m = model.coef_[0]
c = model.intercept_

future_weeks = np.array([7, 9]).reshape(-1, 1)
predicted_sales = model.predict(future_weeks)

x_range = np.arange(1, 10, 0.1).reshape(-1, 1)
y_range = model.predict(x_range)

plt.scatter(xi, yi, color = 'blue', label = 'Actual Sales')
plt.plot(x_range, y_range, color = 'red', label = f'Regression line: y = {m:.2f}x + {c:.2f}')
plt.scatter(future_weeks, predicted_sales, color = 'green', marker = 'o',
            label = 'Predicted Sales (Weeks 7 and 9)')

plt.xlabel('Weeks')
plt.ylabel('Sales')
plt.title('Weekly Sales Prediction using Linear Regression')
plt.legend()
plt.grid(True)
```

```
plt.show()
print(f"Equation of the regression line : y = {m:.2f}x + {c:.2f}")
print(f"Predicted sales for week 7 : {predicted_sales[0]:.2f}")
print(f"Predicted sales for week 9 : {predicted_sales[1]:.2f}")
```

O/P



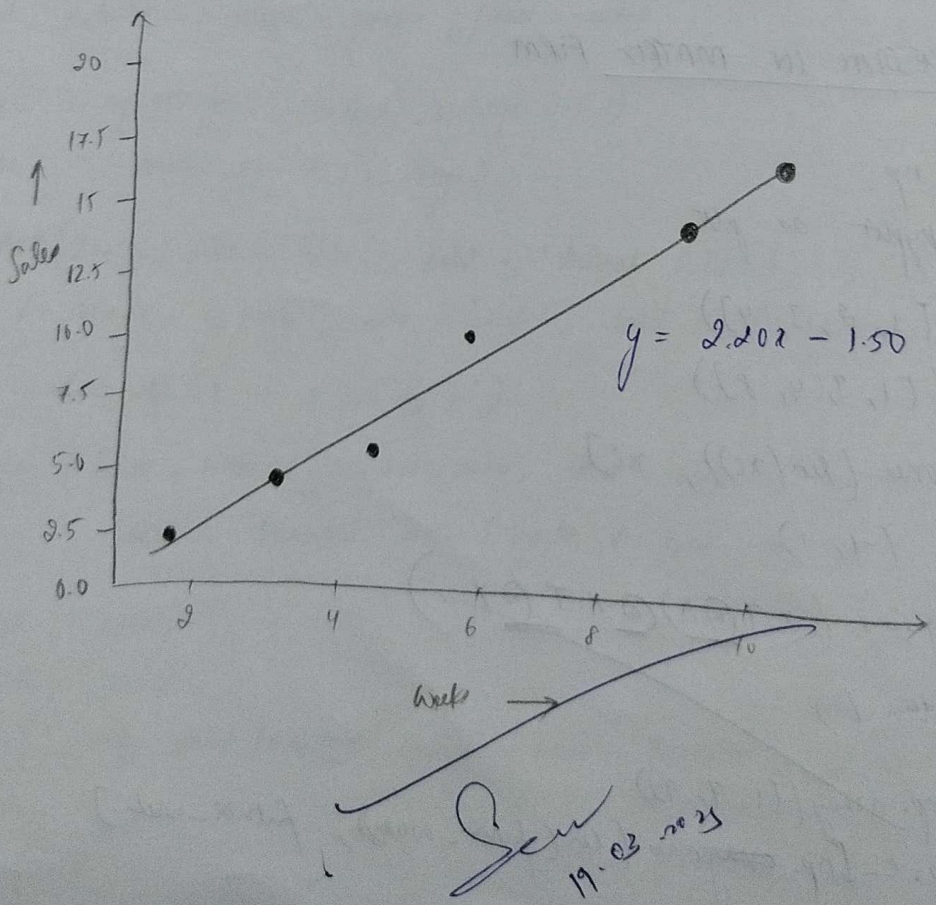
Equation of regression line : $y = 0.66x + 0.54$
Predicted sales for week 7 : 5.16
week 9 : 6.48

(2) LINEAR REGRESSION IN MATRIX FORM

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

xi = np.array([1, 2, 3, 4])
yi = np.array([1, 3, 4, 8])

X = np.c_[np.ones(len(xi)), xi]
Y = yi.reshape(-1, 1)

theta = np.linalg.inv(X.T @ X) @ X.T @ Y

c, m = theta.flatten()

future_week = np.array([1, 7, 9])
X_future = np.c_[np.ones(len(future_week)), future_week]
predicted_sale = X_future @ theta
```

```
x_range = np.linspace(1, 10, 100)
y_range = c + m * x_range

plt.scatter(xi, yi, color = 'blue', label = 'Actual Sales')
plt.plot(x_range, y_range, color = 'red', label = f'Regression Line:
                y = {m:.2f}x + {c:.2f}")
plt.scatter([7,9], predicted_sales[1:], color = 'green',
            marker = 'o', label = 'Predicted_Sales')

plt.xlabel('Weeks')
plt.ylabel('Sales')
plt.title('Weekly Sale Prediction using Matrix Approach')
plt.legend()
plt.grid(True)
plt.show()
```

6) Equation of the regression line $y = 2.20x - 1.50$.



19.03.2025

02/04

LAB - 03

LOGISTIC REGRESSION

Q1. Consider a binary classification problem where we want to predict whether a student will pass or fail.

Given $a_0 = -5$ (intercept)

$a_1 = 0.8$ (coefficient)

(a) Write the logistic regression equation for this problem.

$$\text{sigmoid}(z) = P(x) = \frac{1}{1+e^{-z}} \qquad z = a_0 + a_1 x.$$

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

(b) Calculate the probability that a student who studies for 7 hours will pass.

$$\text{Probability}(x/\text{pass}) = \frac{1}{1+e^{5-0.8(7)}} = \frac{1}{1+e^{-0.6}}$$

$$= 0.6457$$

(c) Determine the predicted class (pass or fail) for this student based on a threshold.

If threshold = 0.5

$P(x = 7) \quad <> 0.5$

Student will ~~fail~~ pass.

Q2. Consider $z = [2, 1, 0]$ for three classes. Apply softmax function

$$\text{softmax}(z_k) = \frac{e^{z_k}}{\sum_{j=1}^{k} e^{z_j}}$$

$$\text{softmax}(z_1) = \frac{e^2}{e^2 + e^1 + e^0} = \frac{7.39}{7.39 + 2.72 + 1} \approx 0.665$$

$$\text{softmax}(z_2) = \frac{e^1}{e^2 + e^1 + e^0} = \frac{2.72}{7.39 + 2.72 + 1} \approx 0.244$$

$$\text{softmax}(z_3) = \frac{e^0}{e^2 + e^1 + e^0} = \frac{1}{7.39 + 2.72 + 1} \approx 0.091$$

Probabilities for three classes = 66.5%, 24.4%, 9.1%.

Q. HR_comma_sep.csv

(i) Which variable did you identify as having a direct and clear impact on employee retention? Why?

Variables such as satisfaction_level, average_monthly_hours, number_project, time_spend_company, salary.

- satisfaction level : strong negative correlation
- average_monthly_hours : strong positive correlation
- number_project : positive correlation
- time_spend_company : strong positive correlation
- salary : strong negative correlation

(ii) What was the accuracy of your logistic regression model?

The accuracy was around (0.79) (79%.

Q. Zoo.csv

(i) Did you perform any data preprocessing steps? If yes, what were they?

① Handling categorial data

class_type → class_mapping

② Splitting dataset

80% training, 20% testing

③ Drop irrelevant columns

Drop animal_name as it is irrelevant.

④ Feature Scaling.

StandardScaler() to standardize numerical features mean = 0, variance = 1

(ii) Were there any missing or inconsistent values in the dataset?

Ans No missing values.

(iii) What does the confusion matrix tell you about the performance of your

The confusion matrix represents how well the model classifies different classes.

Diagonal values : Correct classifications

Other values : Incorrect classification

(iv) Which class type were most frequently misclassified?

Ans Class 3, 5, and 6 were misclassified.

Possible reasons ① Feature similarity

② Small sample size

12/02/25

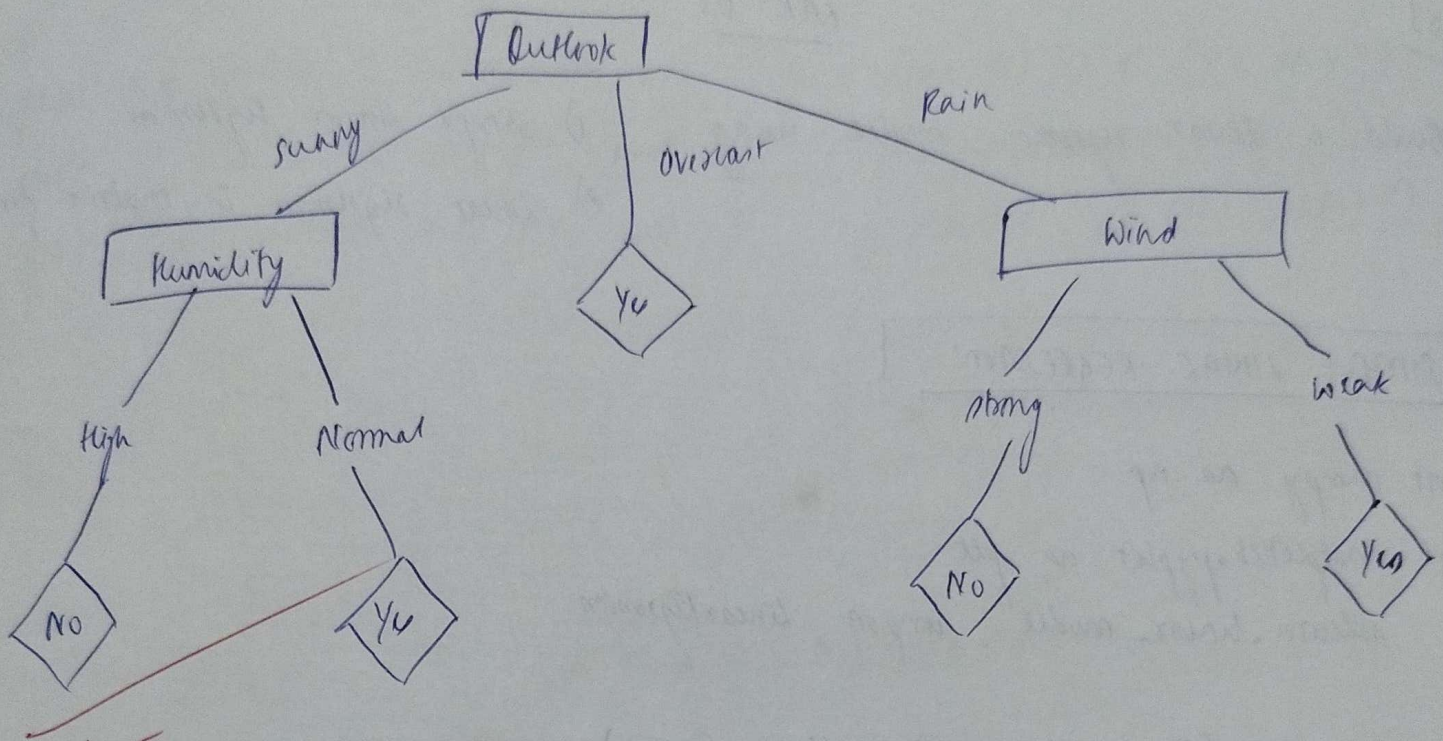
LAB - 02

ID3 (Iterative Dichotomizer 3)

```
import pandas as pd
import math
from collections import Counter

df = pd.read_csv("/content/id3.csv")
df.dropna(inplace = True)

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[target_attribute].unique()) == 1:
        return data[target_attribute].iloc[0]

    if len(features) == 0:
        return data[target_attribute].mode()[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 subset.empty:
        tree[best_feature][value] = data[target_attribute].mode()[0]
    else:
        tree[best_feature][value] = id3(subset, feature, target_attribute)

return tree.

target_attribute = 'label'
features = [col for col in df.columns if col != target_attribute]

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

o/p {'outlook': {'sunny': {'humidity': {'high': 'no', 'normal': 'yes'}},
'overcast': 'yes',
'rainy': {'wind': {'weak': 'yes', 'strong': 'no'}}}}


Outlook
sunny
Overcast
Rain
Humidity
Yes
Wind
High
Normal
strong
weak
No
Yes
No
Yes

02/04/25

LAB - 04

KNN Classification

Q1. Consider the following dataset $k = 3$

Test data (X, 35, 100) as (Person, Age, Salary k)

Person	Age	Salary k	Target	Distance	Rank
A	18	50	N	52.81	5
B	23	55	N	46.52	4
C	24	70	N	31.96	2
D	41	60	Y	40.45	3
E	43	70	Y	31.06	1
F	38	40	Y	60.08	6
X	35	100	?		

For person A, Distance $= \sqrt{(35-18)^2 + (100-50)^2}$

$= 52.81$

For person B, Distance $= \sqrt{(35-23)^2 + (100-55)^2} = 46.52$

person C, distance $= \sqrt{(35-24)^2 + (100-70)^2} = 31.96$

person D, $= \sqrt{(35-41)^2 + (100-60)^2} = 40.45$

For $k = 1$, target $= Y$

$k = 2$ target $= N$

$k = 3$ target $= Y$

mode $(k_1, k_2, k_3) = Y$.

∴ For test data (X, 35, 100) target $= Y$

Q2. For Iris dataset,

How to choose K value? Demonstrate using accuracy rate and error rate?

Ans To determine best value of K, check ① accuracy rate
② error rate

Best K value : 6

Highest accuracy rate b/w K = 1 to 20 was 96.67 %.

Lowest error rate : 3.33%

Diabetes dataset

What is the purpose of feature scaling?

- Since it is distance based algorithm, if dataset contains features with different scales, the model may be biased towards features with larger numerical value.
- StandardScaler() was applied mean = 0
S.D = 1

Sen
02-04-2024

16/04

LAB

RANDOM FOREST CLASSIFIER

Implement the Random Forest classifier using scikit-learn -

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

data = load_iris()
X = data.data
y = data.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 42)

rf_classifier = RandomForestClassifier(n_estimators = 100, random_state = 42)
rf_classifier.fit(X_train, y_train)
y_pred = rf_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")
```

16.04.24

07/05 For "iris.csv" dataset –

Qr. What is the best accuracy score and confusion matrix of the classifier you observed and using how many trees ?

Soln Best observed accuracy score = 1.00 (100 %)

Perfect accuracy was achieved using number of trees

n_estimators : 1.

Confusion matrix.

Actual \ Predicted	setosa	versicolor	virginica
setosa	10	0	0
versicolor	0	9	0
virginica	0	0	11

07/05/25

LAB - 08

Boosting ensemble method

Adaboost algorithm

CGPA	Interactiveness	Practical knowledge	Communication skill	Job Profile
>= 9	Yes	Good	Good	Yes
< 9	No	Good	Moderate	Yes
>= 9	No	Average	Moderate	No
< 9	No	Average	Good	No
>= 9	Yes	Good	Moderate	Yes
>= 9	Yes	Good	Moderate	Yes

Step 1 : Consider 4 decision stumps —
DS (CGPA), DS (Interactiveness), DS (Practical knowledge), DS (Communication skill)

Step 1 : Assign initial weight for each item = 1/6

Step 2 : Iterate for each weak classifier —

Decision stump for CGPA.

	CGPA	Predicted Job offer	Actual Job offer	Weight
①	>= 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

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

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

$$= 0.347$$

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

$$Z_{CGPA} = 0.9428$$

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

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

I D.S for Interactiveness

Interactiveness	Predicted	Actual	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

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

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

$= 0.5490$

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

$= 0.5490$

$wt\ (d_j)_{i+1} = \frac{0.1249 * e^{-0.549}}{0.816} = 0.0892$

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

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

III DS for practical knowledge

- No misclassification

IV DS for communication skills

V

$\alpha_{CGPA} = 0.547$	$\alpha_{interact} = 0.5470$	$\alpha_{comm} = -0.5455$	Final prediction
Yes	Yes	Yes	Yes
No	No	No	No
Yes	No	No	Yes
No	No	Yes	No
Yes	Yes	No	Yes
Yes	Yes	No	Yes.

Income.csv

Q1. What is the best accuracy score and confusion matrix of the classifier?

Ans Best accuracy score = 0.8140

No. of estimators = 73

Confusion matrix (n_estimators = 10)

	0	1
0	6782	632
1	1144	1211

LAB - 09

K mean clustering

07/05/2025

Q. For the given data, compute 2 cluster using K mean algorithm for clustering where initial cluster centers are (1.0, 1.0) and (5.0, 7.0)

Ans No. of cluster, $k = 2$.

centroid C1 = (1.0, 1.0)

centroid C2 = (5.0, 7.0)

Record Number	close to C1	close to C2	Assign to cluster
R1 (1.0, 1.0)	0.0	7.21	cluster1
R2 (1.5, 2.0)	1.12	6.12	cluster1
R3 (3.0, 4.0)	3.61	3.61	cluster1
R4 (5.0, 7.0)	7.21	0.0	cluster2
R5 (3.5, 5.0)	4.12	2.5	cluster2
R6 (4.5, 5.0)	5.31	2.06	cluster2
R7 (3.5, 4.5)	4.30	2.92	cluster2

cluster 1 = { R1, R2, R3 }

cluster 2 = { R4, R5, R6, R7 }.

New centroids

$$C1 = \left(\frac{1.0 + 1.5 + 3.0}{3}, \frac{1.0 + 2.0 + 4.0}{3}\right)$$

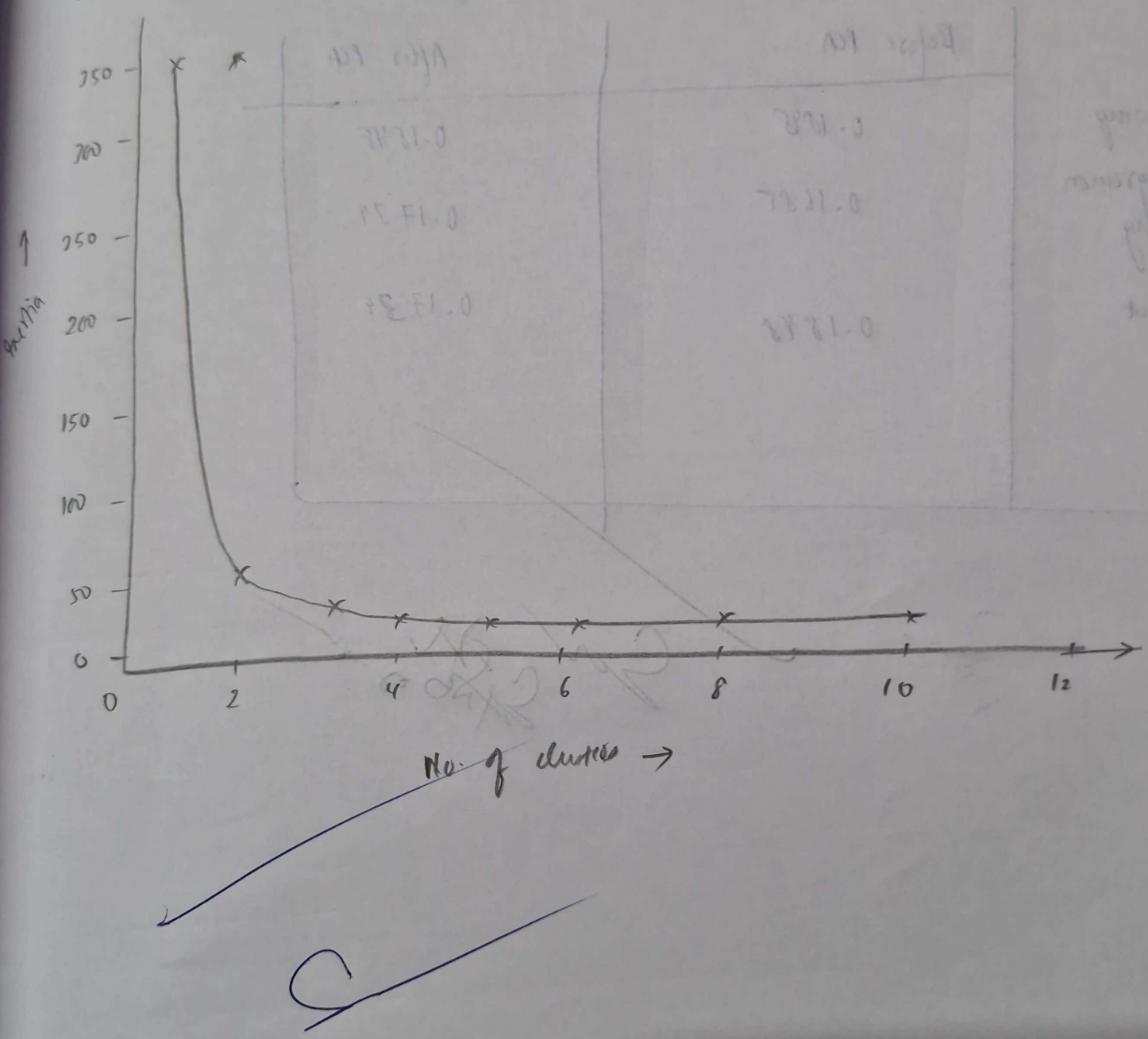
$$= (1.83, 2.33)$$

$$C2 = (4.12, 5.37)$$

Ques. 07

Draw the elbow plot. What was the optimal k value obtained.

Optimal value of k = 3



07/05/25

LAB - 10

PCA

heart.csv

Report the accuracy score before and after applying PCA

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