VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Machine Learning (23CS6PCMAL)

Submitted by

Varsha Prasanth (1BM22CS321)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
Sep-2024 to Jan-2025

B.M.S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "Machine Learning (23CS6PCMAL)" carried out by **Varsha Prasanth (1BM22CS321),** who is a bonafide student of **B.M.S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements in respect of an Machine Learning (23CS6PCMAL) work prescribed for the said degree.

Ramya K M	Dr. Kavitha Sooda
Assistant Professor	Professor & HOD
Department of CSE , BMSCE	Department of CSE , BMSCE

Index

Github Link:

https://github.com/Varsha-Pr/ML-Lab

Program 1

Write a python program to import and export data using Pandas library functions

Code:

```
import pandas as pd
data = {
'Name': ['Alice', 'Bob', 'Charlie', 'David'],
'Age': [25, 30, 35, 40],
'City': ['New York', 'Los Angeles', 'Chicago', 'Houston']
}
df = pd.DataFrame(data)
print("Sample data:")
print(df.head())
from sklearn.datasets import load_iris
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['target'] = iris.target
print("Sample data:")
print(df.head())
file_path = 'data.csv'
df = pd.read_csv(file_path)
print("Sample data:")
print(df.head())
print("\n")
file_path = 'mobiles-dataset-2025.csv'
df = pd.read_csv(file_path, encoding='latin-1') # or 'cp1252' or other suitable encoding
print("Sample data:")
print(df.head())
import pandas as pd

data = {
'USN': ['IS001','IS002','IS003','IS004','IS005'],
'Name': ['Alice', 'Bob', 'Charlie', 'David','Eve'],
'Marks': [25, 30, 35, 40,45]
}

df = pd.DataFrame(data)
print("Sample data:")
print(df.head())
from sklearn.datasets import load_diabetes
iris = load_diabetes()
df = pd.DataFrame(iris.data, columns=iris.feature_names)

print("Sample data:")
print(df.head())
```

```
file_path = 'sample_sales_data.csv'
df = pd.read_csv(file_path)
print("Sample data:")
print(df.head())
print("\n")

df = pd.read_csv("/content/dataset-of-diabetes .csv",encoding='latin-1')
print("Sample data:")
print(df.head())
print("\n")


df =pd.read_csv('sample_sales_data.csv')
print("Sample data:")
print(df.head())

df.to_csv('output.csv',index=False)
print("Data saved to output.csv")
sales_df =pd.read_csv('sample_sales_data.csv')
print("Sample data:")
print(sales_df.head())
sales_by_region =sales_df.groupby('Region')['Sales'].sum()
print("\nTotal sales by region:")
print(sales_by_region)
best_selling_products
=sales_df.groupby('Product')['Quantity'].sum().sort_values(ascending=False) print("\nBest-selling
products by quantity:")
print(best_selling_products)
sales_by_region.to_csv('sales_by_region.csv')
best_selling_products.to_csv('best_selling_products.csv')
print("Data saved to sales_by_region.csv and best_selling_products.csv")

import yfinance as yf
import matplotlib.pyplot as plt
tickers = ["RELIANCE.NS", "TCS.NS", "INFY.NS"]
data = yf.download(tickers, start="2022-10-01", end="2023-10-01",
group_by='ticker')
print("First 5 rows of the dataset:")
print(data.head())
print("\nShape of the dataset:")
print(data.shape)
print("\nColumn names:")
print(data.columns)
print("\n")
reliance_data = data['RELIANCE.NS']
print("\nSummary statistics for Reliance Industries:")
print(reliance_data.describe())
reliance_data['Daily Return'] = reliance_data['Close'].pct_change()
print("\n")
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
reliance_data['Close'].plot(title="Reliance Industries - Closing Price")
```

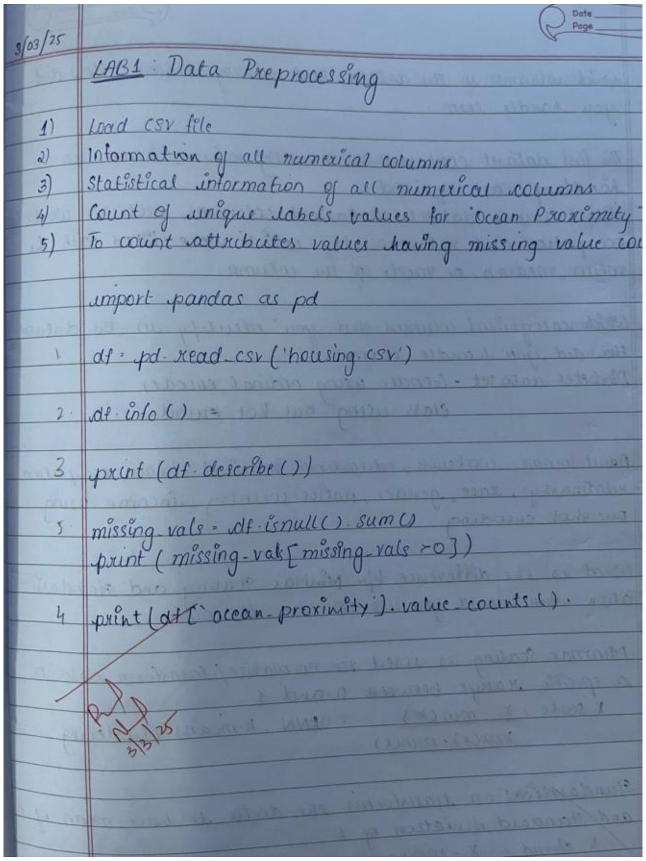
```
plt.subplot(2, 1, 2)
reliance_data['Daily Return'].plot(title="Reliance Industries - Daily Returns", color='orange')
plt.tight_layout()
plt.show()
reliance_data.to_csv('reliance_stock_data.csv')

tickers = ["HDFCBANK.NS", "ICICI.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())
print("\nShape of the dataset:")
print(data.shape)
print("\nColumn names:")
print(data.columns)
print("\n")
reliance_data = data['HDFCBANK.NS']
print("\nSummary statistics for Reliance Industries:")
print(reliance_data.describe())
reliance_data['Daily Return'] = reliance_data['Close'].pct_change()
print("\n")
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
reliance_data['Close'].plot(title="HDFC Industries - Closing Price")
plt.subplot(2, 1, 2)
reliance_data['Daily Return'].plot(title="HDFCIndustries - Daily Returns", color='red')
plt.tight_layout()
plt.show()
reliance_data.to_csv('hdfc_stock_data.csv')
print("\nhdfc stock data saved to 'hdfc_stock_data.csv'.")

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())
print("\nShape of the dataset:")
print(data.shape)
print("\nColumn names:")
print(data.columns)
print("\n")
reliance_data = data['ICICIBANK.NS']
print("\nSummary statistics for ICICI Industries:")
print(reliance_data.describe())
reliance_data['Daily Return'] = reliance_data['Close'].pct_change()
print("\n")
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
reliance_data['Close'].plot(title="ICICI Industries - Closing Price")
plt.subplot(2, 1, 2)
reliance_data['Daily Return'].plot(title="ICICI Industries - Daily Returns", color='BLACK')
plt.tight_layout()
```

```
plt.show()
reliance_data.to_csv('icici_stock_data.csv')
print("\nicici stock data saved to 'icici_stock_data.csv'.")


tickers = ["HDFCBANK.NS", "ICICI.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())
print("\nShape of the dataset:")
print(data.shape)
print("\nColumn names:")
print(data.columns)
print("\n")
reliance_data = data['KOTAKBANK.NS']
print("\nSummary statistics for Reliance Industries:")
print(reliance_data.describe())
reliance_data['Daily Return'] = reliance_data['Close'].pct_change()
print("\n")
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
reliance_data['Close'].plot(title="KOTAK Industries - Closing Price")
plt.subplot(2, 1, 2)
reliance_data['Daily Return'].plot(title="kotak Industries - Daily Returns", color='red')
plt.tight_layout()
plt.show()
reliance_data.to_csv('kotak_stock_data.csv')
print("\nkotak stock data saved to 'kotak_stock_data.csv'.")
```

<u>**Program 2**</u>

Demonstrate various data pre-processing techniques for a given dataset

Screenshot:


3/03/25

LAB1 : Data Preprocessing

1) Load csv file
2) Information of all numerical columns
3) Statistical information of all numerical columns
4) Count of unique labels values for "Ocean Proximity"
5) To count attributes values having missing value co

import pandas as pd

1 df = pd.read_csv('housing.csv')

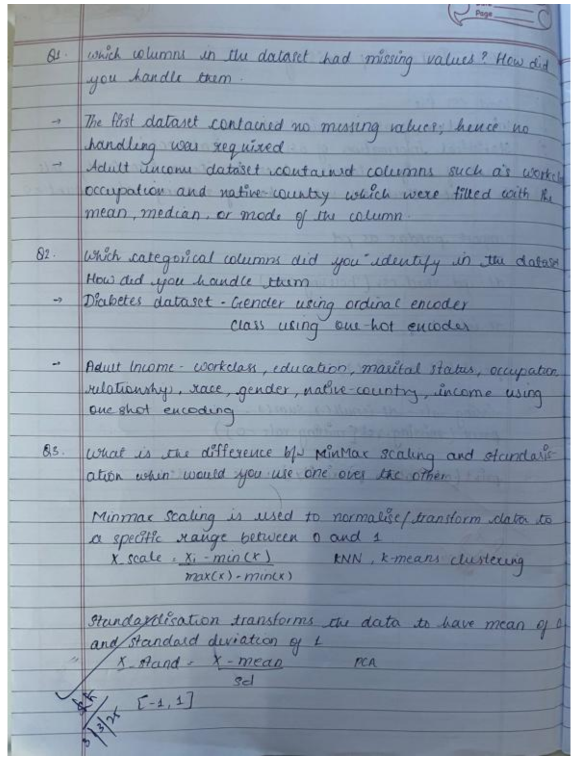
2. df.info()

3 print(df.describe())

5 missing_vals = df.isnull().sum()
print(missing_vals[missing_vals > 0])

4 print(df['ocean_proximity'].value_counts().

3/3/25


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

→ The first dataset contained no missing values; hence no handling was required
→ Adult income dataset contained columns such as workcl occupation and native-country which were filled with the mean, median, or mode of the column.

Q2. Which categorical columns did you identify in the datase How did you handle them
→ Diabetes dataset - Gender using ordinal encoder
Class using one-hot encoder

→ Adult Income - workclass, education, marital status, occupation relationship, race, gender, native-country, income using one shot encoding

Q3. What is the difference b/w MinMax scaling and standardisation when would you use one over the other

Minmax Scaling is used to normalise/transform data to a specific range between 0 and 1
X_scale = (Xi - min(x)) / (max(x) - min(x))   KNN, k-means clustering

Standardisation transforms the data to have mean of 0 and standard deviation of 1
X_stand = (X - mean) / sd   PCA
[-1, 1]

3/3/25

Code:

```
from google.colab import files
diabetes=files.upload()

from google.colab import files
adult_income=files.upload()

df1=pd.read_csv("Dataset of Diabetes .csv")
df1.head()

df2=pd.read_csv("adult.csv")
df2.head()

df1.info()
df2.info()
```

```
df1.describe()
df2.describe()

missing_values1 = df1.isnull().sum()
print(missing_values1)
missing_values2 = df2.isnull().sum()
print(missing_values2)

df1['Gender'] = df1['Gender'].replace('f', 'F')
ordinal_encoder = OrdinalEncoder(categories=[["F", M"]])
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)
df1 = pd.concat([df1, encoded_df], axis=1)
df1.drop("CLASS", axis=1, inplace=True)
df1.drop("Gender", axis=1, inplace=True)
print(df2.head())
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder
df_copy2 = df2
ordinal_encoder = OrdinalEncoder(categories=[["Male","Female"]])
df_copy2["Gender_Encoded"] =
ordinal_encoder.fit_transform(df_copy2[["gender"]])
print(df_copy2[["gender","Gender_Encoded"]])

onehot_encoder = OneHotEncoder()
encoded_data =
onehot_encoder.fit_transform(df2[["occupation","workclass","education","marital-
status","relationship","race","n ative-country","income"]])
encoded_array = encoded_data.toarray()
encoded_df =
pd.DataFrame(encoded_array,
columns=onehot_encoder.get_feature_names_out(["occupation","workclass","education","marital-
status","relatio nship","race","native-country","income"]))
df_encoded = pd.concat([df_copy2, encoded_df], axis=1)

df_encoded.drop("gender", axis=1, inplace=True)
df_encoded.drop("occupation", axis=1, inplace=True)
df_encoded.drop("workclass", axis=1, inplace=True)
df_encoded.drop("education", axis=1, inplace=True)
df_encoded.drop("marital-status", axis=1, inplace=True)
df_encoded.drop("relationship", axis=1, inplace=True)
df_encoded.drop("race", axis=1, inplace=True)
```

```
df_encoded.drop("native-country", axis=1, nplace=True)
df_encoded.drop("income", axis=1, inplace=True)
print(df_encoded. head())

normalizer = MinMaxScaler()
df_encoded[["fnlwgt","educational-num","capital-gain","capital-loss","hours-per-week"]] =
normalizer.fit_transform(df_encoded[["fnlwgt","educational-num","capital-gain","capital-loss","hours-per-
week"]
])
df_encoded.head()
normalizer = MinMaxScaler()
df1[["No_Pation","AGE","Urea","Cr" , "HbA1c" , "Chol","TG","HDL","LDL","VLDL","BMI"]] =
normalizer.fit_transform(df1[["No_Pation","AGE","Urea","Cr" , "HbA1c" ,
"Chol","TG","HDL","LDL","VLDL","BMI"]])
df1.head()
```

Program 3

Implement Linear and Multi-Linear Regression algorithm using appropriate dataset

Screenshot:

classmate

Date ____
Page ____

LAB 2 :

Solve the linear regression of the data of the work and produc sales

x_i work	y_i (sales in thousands)
1	2
2	4
3	5
4	9

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix} \quad Y = \begin{bmatrix} 2 \\ 4 \\ 5 \\ 9 \end{bmatrix}$$

$$\beta = ((X^T X)^{-1} X^T) Y$$

$X^T X =$

$$(X^T X)^{-1} = \begin{bmatrix} 1.5 & -0.5 \\ -0.5 & 0.2 \end{bmatrix}$$

$$(X^T X)^{-1} X^T = \begin{bmatrix} 1.5 & -0.5 \\ -0.5 & 0.2 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0.5 & 0 & -0.5 \\ -0.3 & -0.1 & 0.1 & 0.3 \end{bmatrix}$$

$$((X^T X)^{-1} X^T) Y = \begin{bmatrix} 1 & 0.5 & 0 & -0.5 \\ -0.3 & -0.1 & 0.1 & 0.3 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ 5 \\ 9 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 2.2 \end{bmatrix} \begin{matrix} \rightarrow \text{intercept} \\ \rightarrow \text{slop} \end{matrix}$$

regression eq^n = $y = a_0 + a_1 x$

$= -0.5 + 2.2x$

Questions :

Q1. Considering the 3 datasets (canada per capita income csv, hiring.c hiring csv), did you perform any data preprocessing steps (ex missing values, scaling etc) if yes, why.

Ans. dataset 1 :

- no missing values were present, so no imputation
- Data was already numeric

2. dataset 2:

missing values in 'experience' column were handled using median

dataset 3:

missing values in experience were handled by filling in median
missing values in test score were replaced by mean.

Q2. For canada per capita income.csv did you visualize the regression line along with the datapoints? What does the plot tell you about the relationship between year and per capita

→ There is positive correlationship between year and per capita income
→ As year increases, per capita income increases, indicating an upward trend.

Q3. For hiring.csv what is the predicted salary for a candidate with 12 years of experience, 10 test scores and 10 interview score?

Soln $87405.80

Q4. For 1000 companies.csv, did you encode variables (ex. state)? If yes, how

Sol Scaling was not done in this model
one-hot was ~~not~~ used
~~label~~ encoding was done using ~~label encoder~~ ( ) since ML models require numerical values.

10/10

10/3/25

Code:

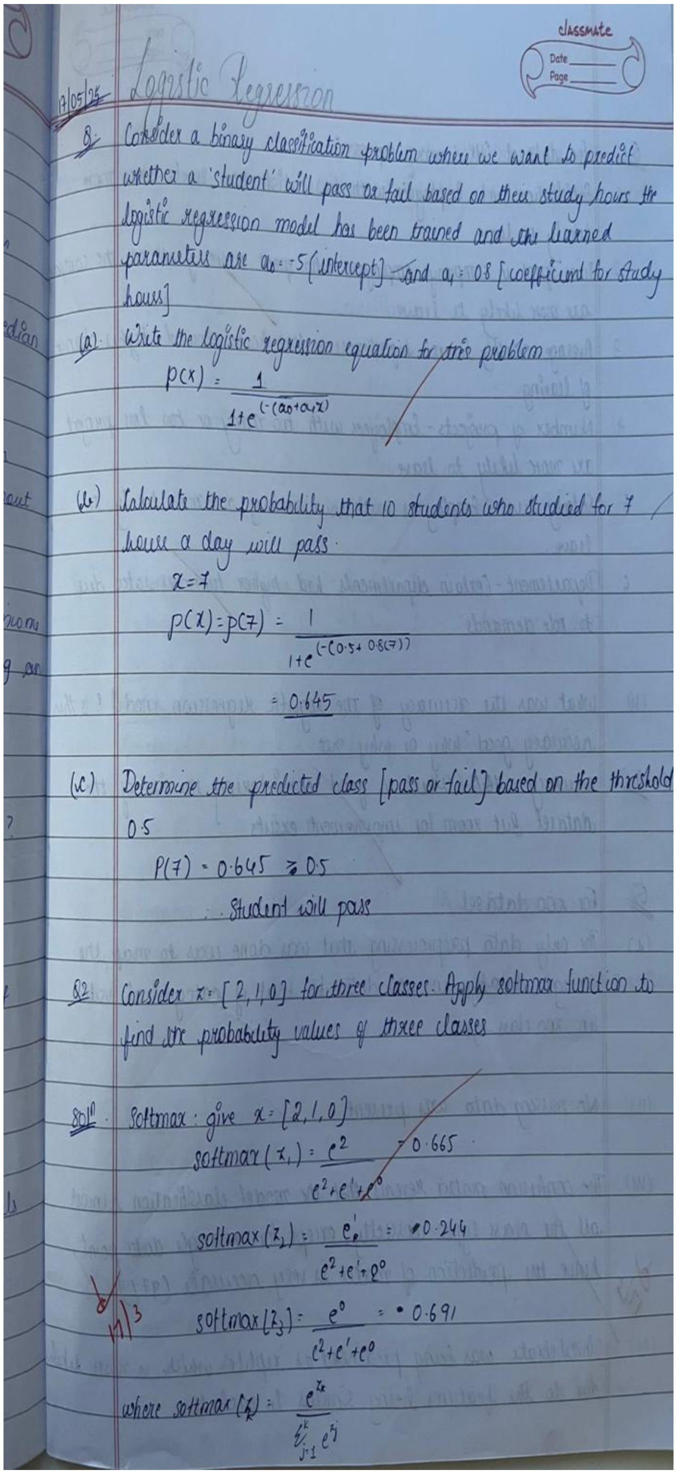
```
from google.colab import files
per_capita_income=files.upload()

from google.colab import files
salary=files.upload()

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from scipy import stats
from sklearn import linear_model

df1=pd.read_csv("canada_per_capita_income.csv")
df1.head()

df2=pd.read_csv("salary.csv")
df2.head()
df2.YearsExperience.median()
df2.YearsExperience =
df2.YearsExperience.fillna(df2.YearsExperience.median()) df2

plt.xlabel("year")
plt.ylabel("per capita income (US$)")
plt.scatter(df1.year, df1['per capita income (US$)'])

plt.xlabel("YearsExperience")
plt.ylabel("Salary")
plt.scatter(df2.YearsExperience, df2.Salary)

reg1 = linear_model.LinearRegression()
reg1.intercept_
reg1.predict([[2020]])

reg2 = linear_model.LinearRegression()
reg2.fit(df2.drop('Salary', axis='columns'), df2['Salary'])
reg2.coef_
reg2.intercept_
reg2.predict([[12]])

from google.colab import files
hiring=files.upload()

from google.colab import files
companies=files.upload()

df3=pd.read_csv("hiring.csv")
```

```
df3.head()

df4=pd.read_csv("1000_Companies.csv")
df4.head()

df3.isnull().sum()
df4.isnull().sum()

df3_copy = df3.copy()
experience_mapping = {'two': 2, 'three': 3, 'five': 5, 'seven': 7, 'ten': 10, 'eleven': 11}
df3_copy['experience'] = df3_copy['experience'].map(experience_mapping)
median_experience = df3_copy['experience'].median()
df3_copy['experience'] = df3_copy['experience'].fillna(median_experience)
df3_copy
df3_copy['test_score(out of 10)'] = df3_copy['test_score(out of 10)'].fillna(df3_copy['test_score(out of
10)'].mean())
reg3 = linear_model.LinearRegression()
reg3.fit(df3_copy.drop('salary($)', axis='columns'),
df3_copy['salary($)']) reg3.coef_
reg3.intercept_
reg3.predict([[2,9,6]])
reg3.predict([[12,10,10]])
ohe = OneHotEncoder(sparse_output=False, handle_unknown='ignore') state_encoded =
ohe.fit_transform(df4[['State']])
state_encoded_df = pd.DataFrame(state_encoded, columns=ohe.get_feature_names_out(['State']))

df4 = pd.concat([df4, state_encoded_df], axis=1).drop(columns=['State'])
print(df4)
reg4 = linear_model.LinearRegression()
reg4.fit(df4.drop('Profit',axis='columns'),df4.Profit)
print(reg4.coef_)
print(reg4.intercept_)
reg4.predict([[91694.48, 515841.3, 11931.24,0,1,0]])
```

Program 4

Build Logistic Regression Model for a given dataset

Screenshot:



17/05/25 Logistic Regression

Q. Consider a binary classification problem where we want to predict whether a 'student' will pass or fail based on their study hours the logistic regression model has been trained and the learned parameters are $a_0 = \cdot 5$ (intercept) and $a_1 = 0.8$ [coefficient for study hours]

(a) Write the logistic regression equation for this problem

$$p(x) = \frac{1}{1+e^{(-(a_0+a_1x))}}$$

(b) Calculate the probability that 10 students who studied for 7 hours a day will pass.

$x = 7$

$$p(x) = p(7) = \frac{1}{1+e^{(-(0.5+0.8(7)))}}$$

$= 0.645$

(c) Determine the predicted class [pass or fail] based on the threshold 0.5

$P(7) = 0.645 \geq 0.5$

$\therefore$ Student will pass

Q2 Consider $z = [2,1,0]$ for three classes. Apply softmax function to find the probability values of three classes

Soln. Softmax: give $z = [2,1,0]$

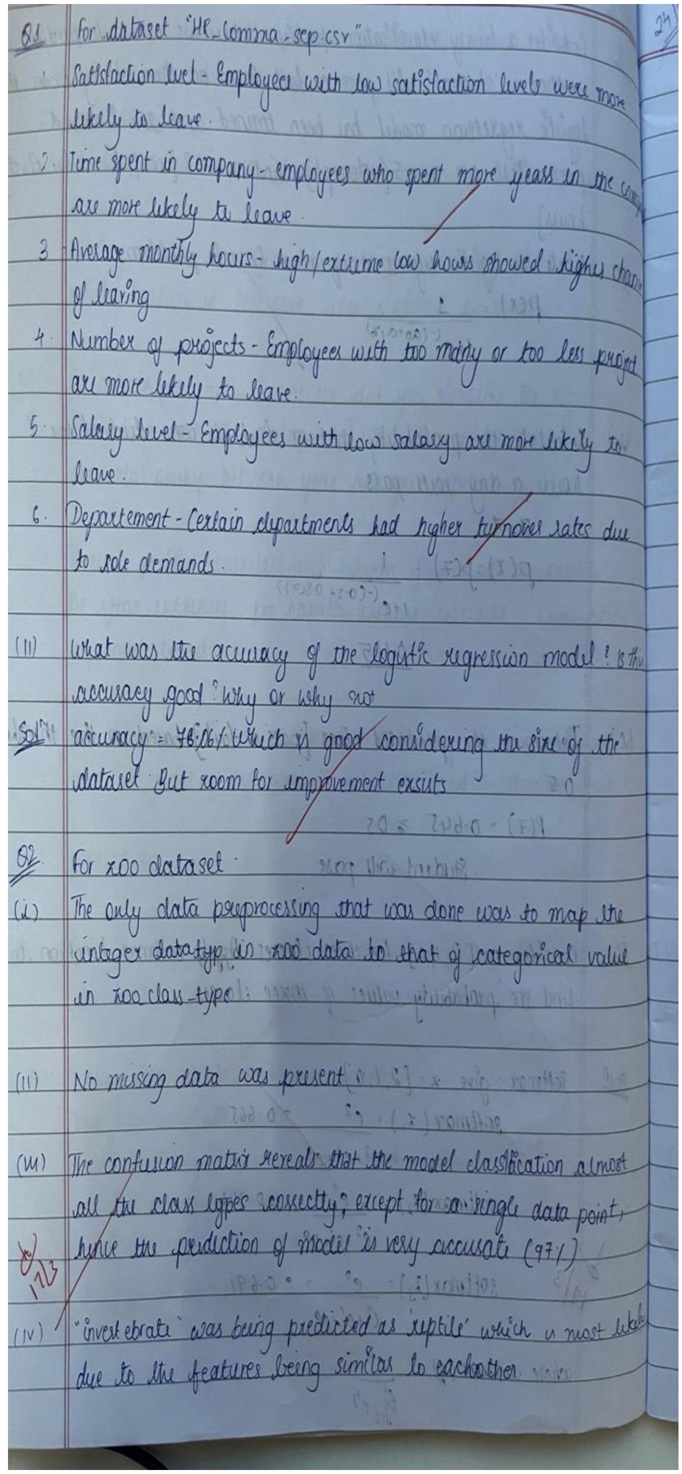
$$\text{softmax}(z_1) = \frac{e^2}{e^2+e^1+e^0} = 0.665$$

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

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

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

17/3



Q1 For dataset "HR_comma_sep.csv"

1. Satisfaction level - Employees with low satisfaction levels were more likely to leave.
2. Time spent in company - employees who spent more years in the company are more likely to leave.
3. Average monthly hours - high/extreme low hours showed higher chance of leaving
4. Number of projects - Employees with too many or too less project are more likely to leave.
5. Salary level - Employees with low salary are more likely to leave.
6. Departement - Certain departments had higher turnover rates due to role demands.

(ii) What was the accuracy of the logistic regression model? Is the accuracy good? Why or why not

Sol. accuracy = 76.06% which is good considering the size of the dataset but room for improvement exists

Q2 For zoo dataset

(i) The only data preprocessing that was done was to map the integer datatype in zoo data to that of categorical value in zoo class_type

(ii) No missing data was present

(iii) The confusion matrix reveals that the model classification almost all the class types correctly, except for a single data point, hence the prediction of model is very accurate (97%)

(iv) "invertebrate" was being predicted as 'reptile' which is most likely due to the features being similar to eachother.

13

Code:

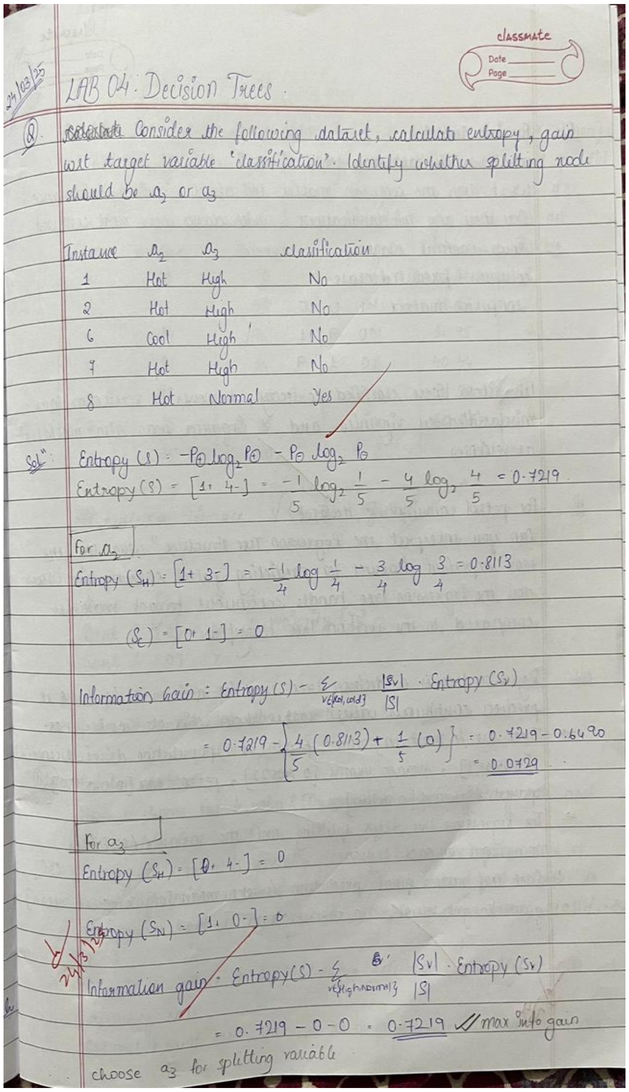
```
from google.colab import files
hr=files.upload()

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from scipy import stats
from sklearn import linear_model
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

df1=pd.read_csv("HR_comma_sep.csv")
df1.head()
df1.isnull().sum()
plt.figure(figsize=(12, 6))
sns.barplot(x='Department', y='left', data=df1)
plt.title('Employee Retention Rate by Department')
plt.xlabel('Department')
plt.ylabel('Proportion of Employees Left')
plt.xticks(rotation=45, ha='right')
plt.show()

ohe = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
department_encoded = ohe.fit_transform(df1[['Department']])
department_encoded_df = pd.DataFrame(department_encoded,
columns=ohe.get_feature_names_out(['Department']))
df1 = pd.concat([df1, department_encoded_df], axis=1)
df1 = df1.drop('Department', axis=1)
ordinal_encoder = OrdinalEncoder(categories=[['low', 'medium', 'high']], dtype=np.int64)
salary_encoded = ordinal_encoder.fit_transform(df1[['salary']])
df1['salary_encoded'] = salary_encoded
df1 = df1.drop('salary', axis=1)
df1.head()

correlation_matrix = df1.corr()
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix of Features')
plt.show()
plt.figure(figsize=(8, 6))
sns.barplot(x='salary_encoded', y='left', data=df1)
plt.title('Impact of Employee Salary on Retention')
plt.xlabel('Salary Level (Encoded)')
plt.ylabel('Proportion of Employees Left')
plt.show()
```

```
df_copy = df1[['number_project', 'average_montly_hours', 'time_spend_company', 'left','salary_encoded',
'satisfaction_level','Work_accident']]
df_copy.head()
X = df_copy.drop('left', axis=1)
y = df_copy['left']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of the Logistic Regression model: {accuracy}")

from google.colab import files
zoodata=files.upload()
zootype=files.upload()

zoo_data = pd.read_csv('zoo-data.csv')
zoo_class = pd.read_csv('zoo-class-type.csv')
merged_data = pd.merge(zoo_data, zoo_class, left_on='class_type', right_on='Class_Number')
merged_data = merged_data.drop(['Animal_Names', 'Number_Of_Animal_Species_In_Class',
'Class_Number','class_type','animal_name'], axis=1)
X = merged_data.drop('Class_Type', axis=1)
y = merged_data['Class_Type']
print(merged_data.head())
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=np.unique(y_test))
disp.plot(cmap="Blues", values_format="d")
plt.title("Confusion Matrix")
plt.show()
```

Program 5

Use an appropriate data set for building the decision tree (ID3) and apply this knowledge to classify a new sample.

Screenshot:



24/03/25

LAB 04: Decision Trees

Q. Consider the following dataset, calculate entropy, gain wrt target variable 'classification'. Identify whether splitting node should be a_2 or a_3

Instance	a_2	a_3	classification
1	Hot	High	No
2	Hot	High	No
6	Cool	High	No
7	Hot	High	No
8	Hot	Normal	Yes

Sol: Entropy (S) = $-P_{\oplus} \log_2 P_{\oplus} - P_{\ominus} \log_2 P_{\ominus}$

Entropy (S) = $[1+, 4-] = -\frac{1}{5}\log_2\frac{1}{5} - \frac{4}{5}\log_2\frac{4}{5} = 0.7219$

For a_2

Entropy (S_H) = $[1+, 3-] = -\frac{1}{4}\log\frac{1}{4} - \frac{3}{4}\log\frac{3}{4} = 0.8113$

(S_C) = $[0+, 1-] = 0$

Information Gain : Entropy (S) $- \sum_{v \in \{hot, cold\}} \frac{|S_v|}{|S|} \cdot$ Entropy (S_v)

$= 0.7219 - \left\{\frac{4}{5}(0.8113) + \frac{1}{5}(0)\right\} = 0.7219 - 0.6490$

$= 0.0729$

For a_3

Entropy (S_H) = $[0+, 4-] = 0$

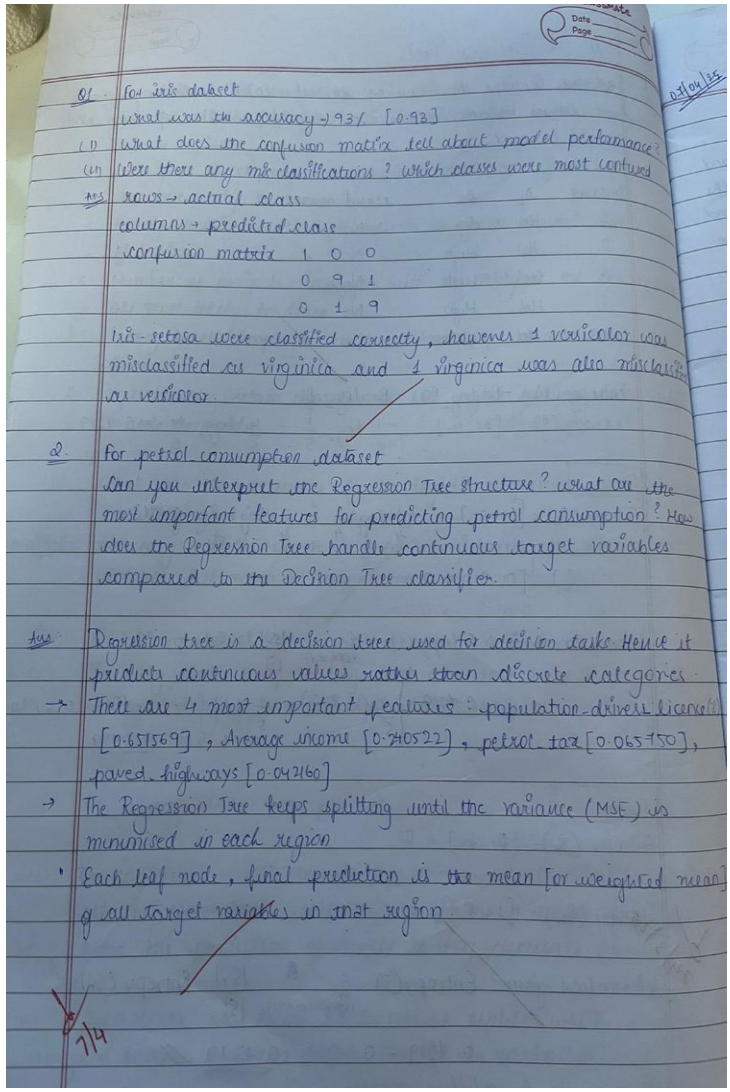
Entropy (S_N) = $[1+, 0-] = 0$

Information gain : Entropy(S) $- \sum_{v \in \{high, normal\}} \frac{|S_v|}{|S|} \cdot$ Entropy (S_v)

$= 0.7219 - 0 - 0 = 0.7219$ // max info gain

choose a_3 for splitting variable

24/3/25



07/04/25

Q1. For iris dataset

what was the accuracy → 93% [0.93]

(i) what does the confusion matrix tell about model performance?

(ii) Were there any misclassifications? which classes were most confused

Ans: rows → actual class

columns → predicted class

confusion matrix
$$\begin{matrix} 1 & 0 & 0 \\ 0 & 9 & 1 \\ 0 & 1 & 9 \end{matrix}$$

Iris-setosa were classified correctly, however 1 versicolor was misclassified as virginica and 1 virginica was also misclassified as versicolor.

Q. For petrol consumption dataset

Can you interpret the Regression Tree structure? what are the most important features for predicting petrol consumption? How does the Regression Tree handle continuous target variables compared to the Decision Tree classifier.

Ans: Regression tree is a decision tree used for decision tasks. Hence it predicts continuous values rather than discrete categories.

→ There are 4 most important features: population-driver licence(%) [0.651569], Average income [0.240522], petrol tax [0.065750], paved highways [0.042160]

→ The Regression Tree keeps splitting until the variance (MSE) is minimised in each region

• Each leaf node, final prediction is the mean [or weighted mean] of all target variables in that region

7/4

Code:

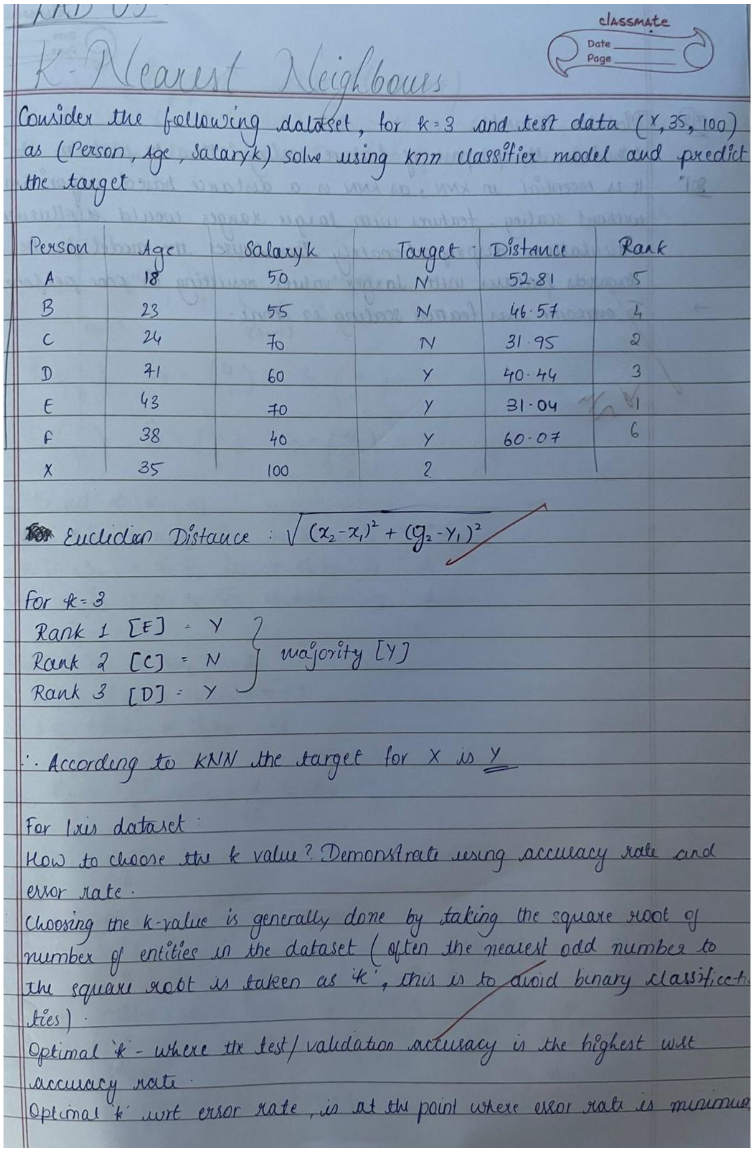
```
from google.colab import files
iris=files.upload()
df1=pd.read_csv("iris.csv")
df1.head()

df1.isnull().sum()

X = df1.drop('species', axis=1)
y = df1['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
clf = DecisionTreeClassifier(criterion='entropy')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
print(classification_report(y_test, y_pred))
plt.figure(figsize=(12, 8))
plot_tree(clf, filled=True, feature_names=X.columns,
class_names=y.unique()) plt.show()

cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=clf.classes_)
cmap = plt.cm.get_cmap('PuBuGn')
disp.plot(cmap=cmap)
plt.show()

drug=files.upload()
df2=pd.read_csv("drug.csv")
df2.head()
df2.isnull().sum()

label_encoders = {}
for column in df2.columns:
    le = LabelEncoder()
    df2[column] = le.fit_transform(df2[column])
    label_encoders[column] = le
X = df2.drop('Drug', axis=1)
y = df2['Drug']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
clf = DecisionTreeClassifier(criterion='entropy')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
print(classification_report(y_test, y_pred))
plt.figure(figsize=(12, 8))
plot_tree(clf, filled=True, feature_names=X.columns, class_names=[str(c) for c in y.unique()])
plt.show()

cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=clf.classes_)
```

```
cmap = plt.cm.Blues
disp.plot(cmap=cmap)
plt.show()

pc=files.upload()
df3=pd.read_csv("petrol_consumption.csv")
df3.head()
df3.isnull().sum()
X = df3.drop('Petrol_Consumption', axis=1)
y = df3['Petrol_Consumption']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
regressor = DecisionTreeRegressor(random_state=42)
regressor.fit(X_train, y_train)
y_pred =
regressor.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Mean Squared Error: {mse:.2f}')
print(f'Root Mean Squared Error:
{rmse:.2f}') print(f'Mean Absolute Error:
{mae:.2f}') print(f'R-squared: {r2:.2f}')
plt.figure(figsize=(30, 30))
plot_tree(regressor, filled=True, feature_names=X.columns, fontsize=10)
plt.show()
```

Program 6

Build KNN Classification model for a given dataset.

Screenshot



K-Nearest Neighbours

Consider the following dataset, for k = 3 and test data (X, 35, 100) as (Person, Age, Salary k) solve using knn classifier model and predict the target

Person	Age	Salary k	Target	Distance	Rank
A	18	50	N	52.81	5
B	23	55	N	46.57	4
C	24	70	N	31.95	2
D	41	60	Y	40.44	3
E	43	70	Y	31.04	1
F	38	40	Y	60.07	6
X	35	100	?		

Euclidean Distance : $\sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$

For k = 3

Rank 1 [E] = Y
Rank 2 [C] = N } majority [Y]
Rank 3 [D] = Y

∴ According to KNN the target for X is Y

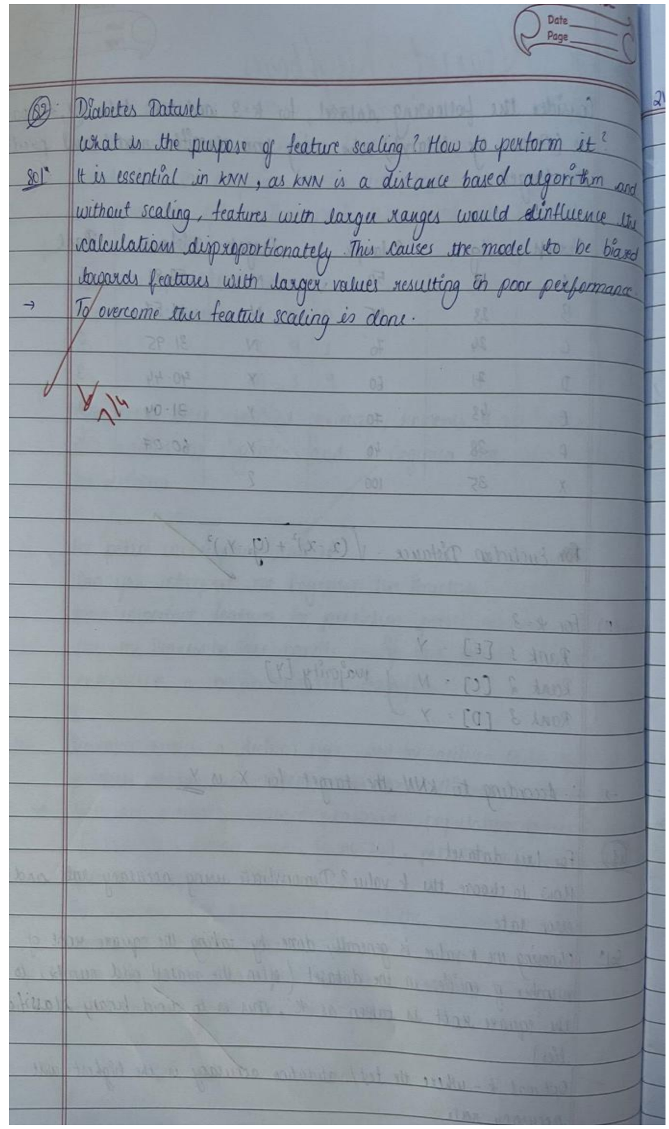
For Iris dataset :

How to choose the k value? Demonstrate using accuracy rate and error rate.

Choosing the k-value is generally done by taking the square root of number of entities in the dataset (often the nearest odd number to the square root is taken as 'k', this is to avoid binary classification ties).

Optimal 'k' - where the test/validation accuracy is the highest wrt accuracy rate.

Optimal 'k' wrt error rate, is at the point where error rate is minimum



(Q2) Diabetes Dataset

What is the purpose of feature scaling? How to perform it?

Sol: It is essential in KNN, as KNN is a distance based algorithm and without scaling, features with larger ranges would influence the calculations disproportionately. This causes the model to be biased towards features with larger values resulting in poor performance.

→ To overcome this feature scaling is done.

Code:

```
from google.colab import files
iris=files.upload()
df1=pd.read_csv("iris (2).csv")
df1.head()
df1.isnull().sum()
X = df1.drop('species', axis=1)
y = df1['species']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
best_k = 1
best_accuracy = 0
for k in range(1,
11):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy for k={k}: {accuracy}, Error Rate for k={k}: {1-accuracy}")if accuracy > best_accuracy:
    best_accuracy = accuracy best_k = k
print(f"Best k value: {best_k}")
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Accuracy Score:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:")
cm = confusion_matrix(y_test, y_pred)
print(cm)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=knn.classes_, yticklabels=knn.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

diabetes=files.upload()
df2=pd.read_csv("diabetes.csv")
df2.head()
df2.isnull().sum()
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df2.drop('Outcome', axis=1))
X_train, X_test, y_train, y_test = train_test_split(X_scaled, df2['Outcome'], test_size=0.2, random_state=42)
best_k = 1
best_accuracy = 0
for k in range(1,
11):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy for k={k}: {accuracy}")
    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_k = k
print(f"Best k value: {best_k}")
```

```
knn = KNeighborsClassifier(n_neighbors=best_k) knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted") plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

heart=files.upload()
df3=pd.read_csv("heart.csv")
df3.head()
df3.isnull().sum()
X = df3.drop('target', axis=1)
y = df3['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
best_k = 1
best_accuracy = 0
for k in range(1,
11):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy for k={k}: {accuracy}, Error Rate for k={k}: {1-accuracy}")
    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_k = k
print(f"Best k value: {best_k}")
knn = KNeighborsClassifier(n_neighbors=optimal_k)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Accuracy Score:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:")
cm = confusion_matrix(y_test, y_pred)
print(cm)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=knn.classes_, yticklabels=knn.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

<u>Program7</u>

Build Support vector machine model for a given dataset

Screenshot:

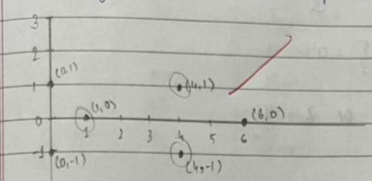
LAB 06:

classmate Date ____ Page ____

21/04/25 SVM (Support Vector Machine)

Points (4,1) (4,-1) & (6,0) belong to the class & points (1,0) (0,1) (0,-1) belong to ve class. Draw an optimal hyperplane.

Sol^n



$\tilde{S}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ $S_2 = \begin{pmatrix} 4 \\ 1 \end{pmatrix}$ $S_3 = \begin{pmatrix} 4 \\ -1 \end{pmatrix}$

$S_1 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$ $\tilde{S}_2 = \begin{pmatrix} 4 \\ 1 \\ 1 \end{pmatrix}$ $\tilde{S}_3 = \begin{pmatrix} 4 \\ -1 \\ 1 \end{pmatrix}$

→ Find α_1 α_2 α_3

$\alpha_1 S_1 \cdot S_1 + \alpha_2 S_2 \cdot S_1 + \alpha_3 S_3 \cdot S_1 = -1$

$\alpha_1 S_1 \cdot S_2 + \alpha_2 S_2 \cdot S_2 + \alpha_3 S_3 \cdot S_2 = 1$

$\alpha_1 S_1 \cdot S_3 + \alpha_2 S_2 \cdot S_3 + \alpha_3 S_3 \cdot S_3 = 1$

$2\alpha_1 + 5\alpha_2 + 5\alpha_3 = -1$

$5\alpha_1 + 18\alpha_2 + 16\alpha_3 = +1$

$5\alpha_1 + 16\alpha_2 + 18\alpha_3 = +1$

$\alpha_1 = \frac{-22}{9}$ $\alpha_2 = \frac{7}{18}$ $\alpha_3 = \frac{7}{18}$

$\Rightarrow \omega = \sum_{i=1}^{3} \alpha_i \tilde{S}_i$

$= \alpha_1 \tilde{S}_1 + \alpha_2 \tilde{S}_2 + \alpha_3 \tilde{S}_3$

$= \frac{-22}{9}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + \frac{7}{18}\begin{pmatrix} 4 \\ 1 \\ 1 \end{pmatrix} + \frac{7}{18}\begin{pmatrix} 4 \\ -1 \\ 1 \end{pmatrix}$

$= \begin{pmatrix} -22/9 \\ 0 \\ -22/9 \end{pmatrix} + \begin{pmatrix} 28/18 \\ 7/18 \\ 7/18 \end{pmatrix} + \begin{pmatrix} 28/18 \\ -7/18 \\ 7/18 \end{pmatrix} = \begin{pmatrix} 2/3 \\ 0 \\ -5/3 \end{pmatrix}$ } w } b

$\therefore \omega = \begin{pmatrix} 2/3 \\ 0 \end{pmatrix}$ $b = \begin{pmatrix} -5/3 \end{pmatrix}$

$\Rightarrow \omega^T x + b = 0$

$\frac{2}{3} x_1 + 0x_2 + \frac{-5}{3} = 0$

$\Rightarrow \frac{2}{3} x_1 - \frac{5}{3} = 0$

$x_1 = 2.5$ or $x_1 = 5/2$

⇒ QUESTIONS:

1 For iris dataset

SVM with linear kernal accuracy : 1

SVM with RBF kernel accuracy : 1

Both linear and RBF kernal gave best performance with accuracy : 1

In the iris dataset small sample, well structured clearly specified linearly seperable.

2. For Letter-recognition

The letters that are the most frequently confused are 'p' with 'r' and 'k' with 'R'.

→ The acc score is 1 reflecting accurate and excellent separability

→ It performs well on letter dataset considerably.

- This dataset is more complex.
- Iris dataset is simpler with less classes/features. This demonstrates SVMs strength in handling high dimensional data.

Code:

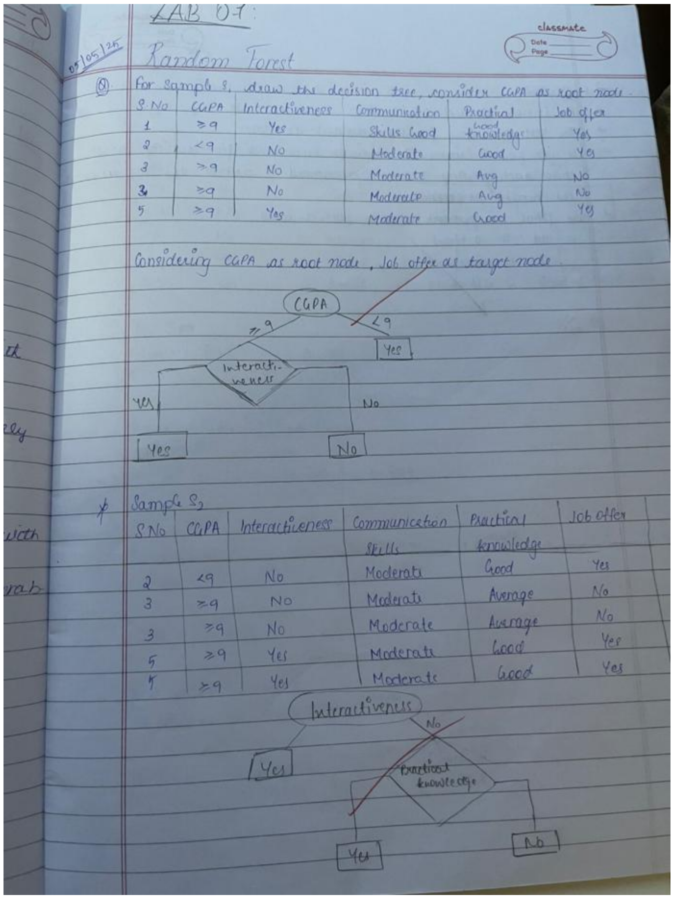
```
from google.colab import files
iris=files.upload()
df1=pd.read_csv("iris (1).csv")
df1.head()
X = df1.drop('species', axis=1)
y = df1['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
rbf_svm = SVC(kernel='rbf')
rbf_svm.fit(X_train, y_train)
rbf_y_pred = rbf_svm.predict(X_test)
print("RBF Kernel SVM:")
print("Accuracy:", accuracy_score(y_test, rbf_y_pred))
cm = confusion_matrix(y_test, rbf_y_pred)
sns.heatmap(cm, annot=True, fmt='d',cmap="Blues")
plt.title('Confusion Matrix for RBF Kernel SVM')
plt.xlabel('Predicted')
plt.ylabel('True') plt.show()
print(classification_report(y_test, rbf_y_pred))
linear_svm = SVC(kernel='linear')
linear_svm.fit(X_train, y_train)
linear_y_pred = linear_svm.predict(X_test)
print("\nLinear Kernel SVM:")
print("Accuracy:", accuracy_score(y_test, linear_y_pred))
cm = confusion_matrix(y_test, linear_y_pred)
sns.heatmap(cm, annot=True, fmt='d',cmap="Blues")
plt.title('Confusion Matrix for Linear Kernel SVM')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
print(classification_report(y_test, linear_y_pred))
letter=files.upload()
df2=pd.read_csv("letter-recognition.csv")
df2.head()
X = df2.drop('letter', axis=1)
y = df2['letter']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
svm_classifier = SVC(kernel='linear', probability=True)
svm_classifier.fit(X_train, y_train)
y_pred =
svm_classifier.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(10,10))
sns.heatmap(cm, annot=True, fmt='d', cmap="Blues")
plt.title('Confusion Matrix for SVM')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
lb = LabelBinarizer()
```

```
lb.fit(y_test)




y_test_lb = lb.transform(y_test)
y_pred_prob =
svm_classifier.predict_proba(X_test) fpr = {}
tpr = {}
thresh ={}
roc_auc = dict()
n_class = y_test_lb.shape[1]
for i in range(n_class):
    fpr[i], tpr[i], thresh[i] = roc_curve(y_test_lb[:,i], y_pred_prob[:,i])
    roc_auc[i] = auc(fpr[i], tpr[i])
plt.plot(fpr[0], tpr[0], linestyle='--',color='orange', label='SVM (AUC = %0.2f)' % roc_auc[0])
plt.title('ROC Curve for Class 0')
plt.xlabel('False Positive
Rate') plt.ylabel('True Positive
rate') plt.legend(loc='best')
plt.show()
print(f"AUC score for class 0: {roc_auc[0]}")
```

Program 8

Implement Random forest ensemble method on a given dataset

Screenshot:



LAB 07:

05/05/25

Random Forest

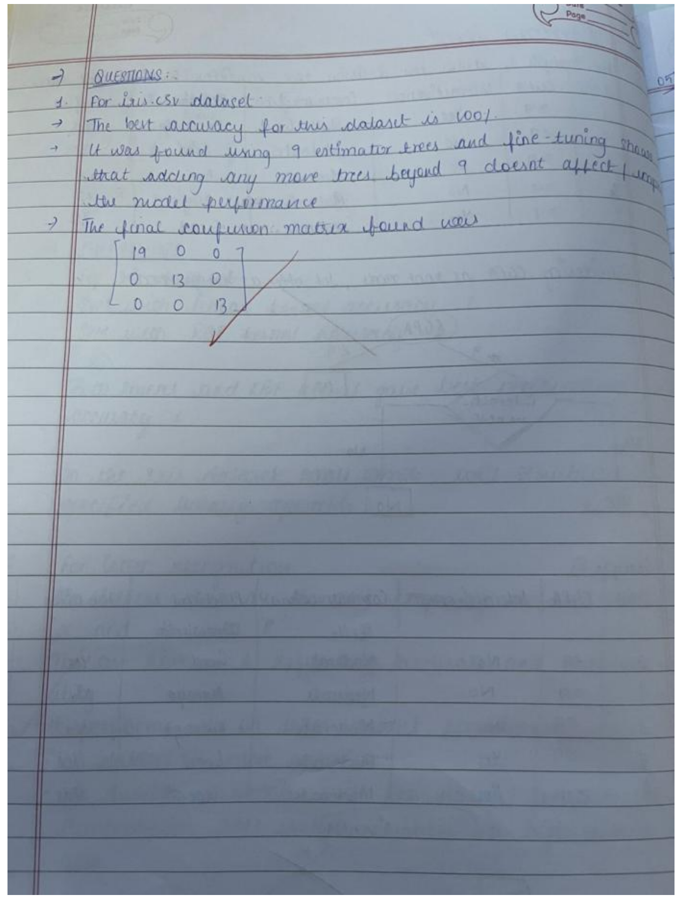
Q. For sample S_1, draw the decision tree, consider CGPA as root node.

S.No	CGPA	Interactiveness	Communication Skills	Practical knowledge	Job offer
1	≥9	Yes	Good	Good	Yes
2	<9	No	Moderate	Good	Yes
3	≥9	No	Moderate	Avg	No
3	≥9	No	Moderate	Avg	No
5	≥9	Yes	Moderate	Good	Yes

Considering CGPA as root node, Job offer as target node

Sample S_2

S No	CGPA	Interactiveness	Communication Skills	Practical knowledge	Job offer
2	<9	No	Moderate	Good	Yes
3	≥9	No	Moderate	Average	No
3	≥9	No	Moderate	Average	No
5	≥9	Yes	Moderate	Good	Yes
5	≥9	Yes	Moderate	Good	Yes



QUESTIONS:

1. For iris.csv dataset

→ The best accuracy for this dataset is 100%.

→ It was found using 9 estimator trees and fine-tuning shows that adding any more trees beyond 9 doesnt affect / improve the model performance

→ The final confusion matrix found was

$$\begin{bmatrix} 19 & 0 & 0 \\ 0 & 13 & 0 \\ 0 & 0 & 13 \end{bmatrix}$$

Code:

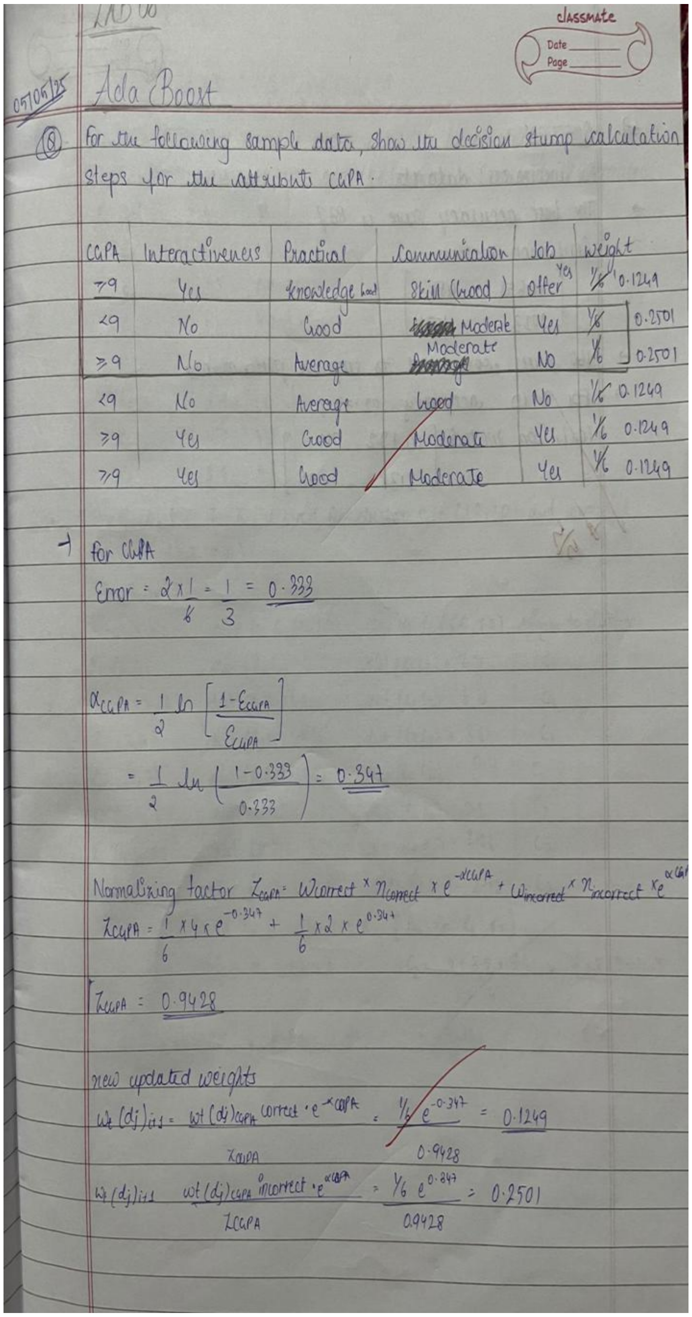
```
from google.colab import files
iris=files.upload()
df1=pd.read_csv("iris (4).csv")
df1.head()
X = df1.drop('species', axis=1)
y = df1['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
rf_classifier = RandomForestClassifier(random_state=0)
rf_classifier.fit(X_train, y_train)
y_pred =
rf_classifier.predict(X_test)
default_accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy with default n_estimators: {default_accuracy}")
best_accuracy = 0
best_n_estimators = 0
for n_estimators in range(1, 101):
    rf_classifier = RandomForestClassifier(n_estimators=n_estimators, random_state=0)
    rf_classifier.fit(X_train, y_train)
    y_pred = rf_classifier.predict(X_test)
```

```
    accuracy = accuracy_score(y_test, y_pred)
    if accuracy > best_accuracy:best_accuracy
    = accuracy best_n_estimators =
    n_estimators
print(f"\nBest accuracy: {best_accuracy} achieved with n_estimators = {best_n_estimators}")
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
        xticklabels=np.unique(y_test), yticklabels=np.unique(y_test))
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

Program 9

Implement Boosting ensemble method on a given dataset

Screenshot:



05/06/25 Ada Boost

Q) For the following sample data, show the decision stump calculation steps for the attribute CGPA.

CGPA	Interactiveness	Practical knowledge	Communication Skill	Job offer	weight
≥9	Yes	Good	Good	Yes	1/6 0.1249
<9	No	Good	Moderate	Yes	1/6 0.2501
≥9	No	Average	Moderate	No	1/6 0.2501
<9	No	Average	Good	No	1/6 0.1249
≥9	Yes	Good	Moderate	Yes	1/6 0.1249
≥9	Yes	Good	Moderate	Yes	1/6 0.1249

→ for CGPA

Error $= 2 \times \frac{1}{6} = \frac{1}{3} = 0.333$

$$\alpha_{CGPA} = \frac{1}{2} \ln \left[\frac{1-E_{CGPA}}{E_{CGPA}} \right]$$

$$= \frac{1}{2} \ln \left(\frac{1-0.333}{0.333} \right) = 0.347$$

Normalizing factor $Z_{CGPA} = W_{correct} \times n_{correct} \times e^{-\alpha CGPA} + W_{incorrect} \times n_{incorrect} \times e^{\alpha CGPA}$

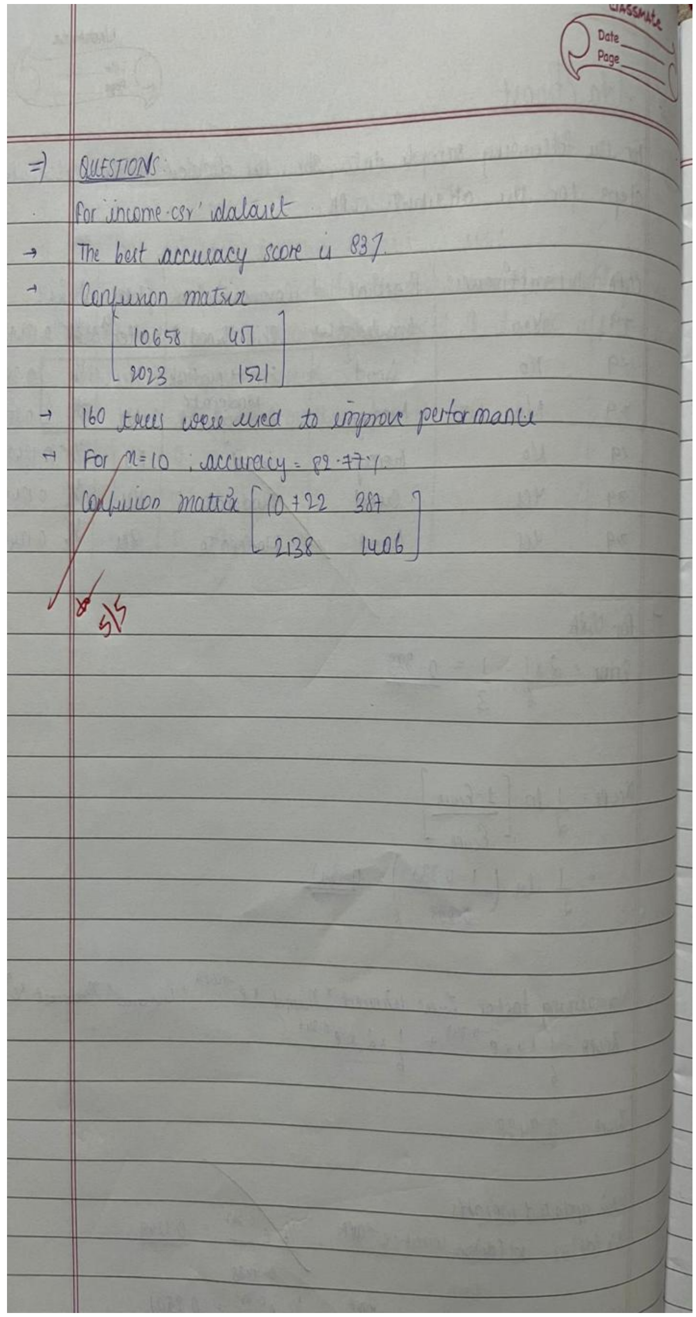
$$Z_{CGPA} = \frac{1}{6} \times 4 \times e^{-0.347} + \frac{1}{6} \times 2 \times e^{0.347}$$

$Z_{CGPA} = 0.9428$

new updated weights

$$wt(d_j)_{i+1} = \frac{wt(d_j)_{CGPA\,correct} \cdot e^{-\alpha CGPA}}{Z_{CGPA}} = \frac{1/6\, e^{-0.347}}{0.9428} = 0.1249$$

$$wt(d_j)_{i+1} = \frac{wt(d_j)_{CGPA\,incorrect} \cdot e^{\alpha CGPA}}{Z_{CGPA}} = \frac{1/6\, e^{0.347}}{0.9428} = 0.2501$$



⇒ QUESTIONS

For 'income.csv' dataset

→ The best accuracy score is 83%.

→ Confusion matrix

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

→ 160 trees were used to improve performance

→ For n=10 ; accuracy = 82.77%

Confusion matrix $\begin{bmatrix} 10722 & 387 \\ 2138 & 1406 \end{bmatrix}$

5/5

Code:

```
from google.colab import files
income=files.upload()
df1=pd.read_csv("income.csv")
df1.head()
X = df1.drop('income_level', axis=1)
y = df1['income_level'] X = pd.get_dummies(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
abc = AdaBoostClassifier(n_estimators=10, random_state=42)
abc.fit(X_train, y_train)
y_pred = abc.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Initial AdaBoost accuracy (10 trees): {accuracy}")
param_grid = {'n_estimators': [50, 100, 150, 200]}
grid_search = GridSearchCV(AdaBoostClassifier(random_state=42), param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)
print(f"Best parameters: {grid_search.best_params_}")
print(f"Best cross-validation score: {grid_search.best_score_}")
best_abc = grid_search.best_estimator_
y_pred_best = best_abc.predict(X_test)
best_accuracy = accuracy_score(y_test,
y_pred_best)
print(f"Accuracy of the best model on the test set: {best_accuracy}")
cm = confusion_matrix(y_test, y_pred_best)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
        xticklabels=['<=50K', '>50K'], yticklabels=['<=50K', '>50K'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Program 10

Build k-Means algorithm to cluster a set of data stored in a .CSV file

Screenshot:

LAB 09

K-MEANS ALGORITHM:

Q. Compute 2 clusters using k-means algorithm for clustering where initial cluster centres are (1.0, 1.0) and (5.0 and 7.0). Execute for 2 iterations

Record no.	A	B
R_1	1.0	1.0
R_2	1.5	2.0
R_3	3.0	4.0
R_4	5.0	7.0
R_5	3.5	5.0
R_6	4.5	5.0
R_7	3.5	4.5

Sol: No. of clusters k = 2, centroid for cluster C_1 = (1.0, 1.0) and centroid for C_2 = (5.0, 7.0)

Iteration 1:

Record no.	close to C_1 (1.0,1.0)	Close to C_2 (5.0, 7.0)	Asign to cluster
R_1 (1,1)	dist (R_1, C_1) = 0.0	dist (R_1 C_2) = 7.21	C_1
R_2 (1.5, 2)	dist (R_2 C_1) = 1.12	dist (R_2 C_2) = 6.12	C_1
R_3 (3,4)	dist (R_3 C_1) = 3.61	dist (R_3 C_2) = 3.61	C_1
R_4 (5,7)	dist (R_4 C_1) = 7.21	dist (R_4 C_2) = 0.0	C_2
R_5 (3.5, 5)	dist (R_5 C_1) = 4.12	dist (R_5 C_2) = 2.5	C_2
R_6 (4.5, 5)	dist (R_6 C_1) = 5.31	dist (R_6 C_2) = 2.06	C_2
R_7 (3.5, 4.5)	dist (R_7 C_1) = 4.30	dist (R_7 C_2) = 2.92	C_2

Cluster 1 {R_1 R_2 R_3} Cluster R_2 {R_4, R_5, R_6, R_7}

$C_1 = \frac{1.0+1.5+3.0}{3}, \frac{1+2+4}{3}$ $\quad C_2 = \frac{5+3.5+3.4}{4}, \frac{7+5+5+4.5}{4}$

$= \underline{1.83}, \underline{2.33}$ $\quad = \underline{4.12}, \underline{5.37}$

Iteration 2

Record no.	Close to C_1 (1.83, 2.33)	Close to C_2 (4.12, 5.37)	Assign to clus
R_1 (1,1)	1.57	5.37	C_1
R_2 (1.5, 2)	0.47	4.27	C_1
R_3 (3,4)	2.04	1.77	C_2
R_4 (5,7)	5.64	1.85	C_2
R_5 (3.5, 5)	3.15	0.72	C_2
R_6 (4.5, 5)	3.78	0.53	C_2
R_7 (3.5, 4.5)	2.74	1.07	C_2

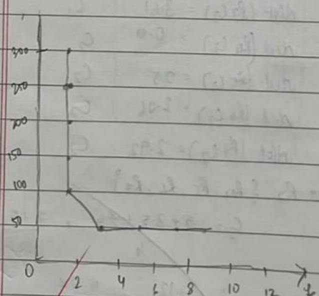
cluster 1 {R_1 R_2} cluster 2 {R_3 R_4 R_5 R_6 R_7}

$C_1 = \frac{1.0+1.5}{2}, \frac{1.0+2.0}{2}$ $\quad C_2 = \frac{3+5+3.5+4.5+3.5}{5}, \frac{4+7+5+5+...}{5}$

$= 1.25, 1.5$ $\quad = \frac{19.5}{5}, \frac{25.5}{5} = \underline{3.9}, \underline{5.1}$

~~opti~~ Questions

optimal number of clusters : 3



Code:

```
from google.colab import files
iris=files.upload()
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from scipy import stats
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

df1=pd.read_csv("iris (4).csv")
df1.head()
df = df1.drop(['sepal_length','sepal_width','species'],axis=1)
scaler = StandardScaler()
scaled_df = scaler.fit_transform(df) wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(scaled_df)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
kmeans = KMeans(n_clusters=3, init='k-means++', max_iter=300, n_init=10, random_state=0)
pred_y = kmeans.fit_predict(scaled_df)
df['cluster'] = pred_y
plt.scatter(df['petal_length'], df['petal_width'], c=df['cluster'])
plt.title('Clusters of Iris Flowers')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.show()
```

Program 11

Implement Dimensionality reduction using Principal Component Analysis (PCA) method.

Screenshot:

LAB 10:

classmate

Date ____ Page ____

PCA:

Q.

Feature	Ex:1	Ex:2	Ex:3	Ex:4
x_1	4	8	13	7
x_2	11	4	5	14

$\lambda_1 = 30.3844$ $\quad e_1 = \begin{bmatrix} 0.5574 \\ -0.8303 \end{bmatrix}$ $\quad e_2 = \begin{bmatrix} 0.8303 \\ 0.5574 \end{bmatrix}$

$\lambda_2 = 6.6151$

Soln: $\bar{x}_1 = \frac{1}{4}(4+8+13+7) = 8$

$\bar{x}_2 = \frac{1}{4}(11+4+5+14) = 8.5$

Step: Covariance matrix

$$S = \begin{bmatrix} cov(x_1 x_1) & cov(x_1 x_2) \\ cov(x_2 x_1) & cov(x_2 x_2) \end{bmatrix}$$

$$cov(x_1 x_1) = \frac{1}{N-1}\left[\sum_{k=1}^{N} (x_{1k} - \bar{x}_1)(x_{1k} - \bar{x}_1)\right]$$

$$= \tfrac{1}{3}\left[(4-8)^2 + (8-8)^2 + (13-8)^2 + (7-8)^2\right]$$

$$= 14$$

$$cov(x_1 x_2) = \tfrac{1}{3}\left[(4-8)(11-8.5) + (8-8)(4-8.5) + (13-8)(5-8.5) + (7-8)(14-8.5)\right]$$

$$= -11$$

$cov(x_2 x_1) = -11 \qquad cov(x_2 x_2) = 23$

$$S = \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

$\lambda_1 = 30.3849 \qquad \lambda_2 = 6.6151$

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \end{bmatrix} = (S - \lambda I)u$$

$$= \begin{bmatrix} 14-\lambda & -11 \\ -11 & 23-\lambda \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} (14-\lambda)u_1 - 11u_2 \\ -11u_1 + (23-\lambda)u_2 \end{bmatrix}$$

$(14-\lambda)u_1 - 11u_2 = 0$

$-11u_1 + (23-\lambda)u_2 = 0$

Date ____ Page ____

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \cdot \frac{u_1}{11} = \frac{u_2}{14-\lambda} = t \qquad U = \begin{bmatrix} 11 \\ 14-\lambda \end{bmatrix}$$

$$e_1 = \begin{bmatrix} 0.5574 \\ -0.8303 \end{bmatrix} \quad e_2 = \begin{bmatrix} 0.8303 \\ 0.5574 \end{bmatrix}$$

$$e^T = \begin{bmatrix} x_{1k} - \bar{x}_1 \\ x_{2k} - \bar{x}_2 \end{bmatrix} = \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} x_{11} - \bar{x}_1 \\ x_{21} - \bar{x}_2 \end{bmatrix}$$

$$= 0.5574(4-8) - 0.8303(11-8.5)$$

$$= -4.30535$$

Components:

Features	Ex1	Ex2	Ex3	Ex4
x_1	4	8	13	7
x_2	11	4	5	14
FPC	-4.3052	3.7361	5.6428	-5.1238

Accuracy Score	Accuracy Score (before PCA)	Accuracy score (After PCA)
logistic regression	0.85	0.83
SVM	0.88	0.86
Random Forest	0.90	0.88

13/15

Code:

```
from google.colab import files
heart=files.upload()

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from scipy import stats
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.decomposition import PCA

df1=pd.read_csv("heart (1).csv")
df1.head()
text_cols = df1.select_dtypes(include=['object']).columns
label_encoder = LabelEncoder()
for col in text_cols:
    df1[col] =
label_encoder.fit_transform(df1[col])
print(df1.head())
X = df1.drop('HeartDisease', axis=1)
y = df1['HeartDisease']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train =
scaler.fit_transform(X_train) X_test =
scaler.transform(X_test)
# Support Vector Machine
svm_model = SVC(kernel='linear', random_state=42)
svm_model.fit(X_train, y_train)
svm_predictions = svm_model.predict(X_test)
svm_accuracy = accuracy_score(y_test, svm_predictions)
print(f"SVM Accuracy: {svm_accuracy}")

# Logistic Regression
lr_model = LogisticRegression(random_state=42)
lr_model.fit(X_train, y_train) lr_predictions = lr_model.predict(X_test)
lr_accuracy = accuracy_score(y_test, lr_predictions)
print(f"Logistic Regression Accuracy: {lr_accuracy}")

# Random Forest
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)
```

```
rf_accuracy = accuracy_score(y_test, rf_predictions)
print(f"Random Forest Accuracy: {rf_accuracy}")


models = {
    "SVM": svm_accuracy,
    "Logistic Regression":
    lr_accuracy, "Random Forest":
    rf_accuracy
}

best_model = max(models, key=models.get)
print(f"\nBest Model: {best_model} with accuracy {models[best_model]}")
pca = PCA(n_components=0.95)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)

svm_model_pca = SVC(kernel='linear', random_state=42)
svm_model_pca.fit(X_train_pca, y_train)
svm_predictions_pca = svm_model_pca.predict(X_test_pca)
svm_accuracy_pca = accuracy_score(y_test, svm_predictions_pca)
print(f"SVM Accuracy (with PCA): {svm_accuracy_pca}")

lr_model_pca = LogisticRegression(random_state=42)
lr_model_pca.fit(X_train_pca, y_train)
lr_predictions_pca = lr_model_pca.predict(X_test_pca)
lr_accuracy_pca = accuracy_score(y_test, lr_predictions_pca)
print(f"Logistic Regression Accuracy (with PCA): {lr_accuracy_pca}")

rf_model_pca = RandomForestClassifier(random_state=42)
rf_model_pca.fit(X_train_pca, y_train)
rf_predictions_pca = rf_model_pca.predict(X_test_pca)
rf_accuracy_pca = accuracy_score(y_test, rf_predictions_pca)
print(f"Random Forest Accuracy (with PCA): {rf_accuracy_pca}")

models_pca = {
    "SVM": svm_accuracy_pca,
    "Logistic Regression": lr_accuracy_pca,
    "Random Forest": rf_accuracy_pca
}

best_model_pca = max(models_pca, key=models_pca.get)
print(f"\nBest Model (with PCA): {best_model_pca} with accuracy {models_pca[best_model_pca]}")
```