VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Machine Learning (23CS6PCMAL)

Submitted by

N Priyanka (1BM22CS167)

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 **N Priyanka (1BM22CS167),** 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.

Pallavi Gb
Assistant Professor
Department of CSE , BMSCE

Dr. Kavitha Sooda
Professor & HOD
Department of CSE , BMSCE

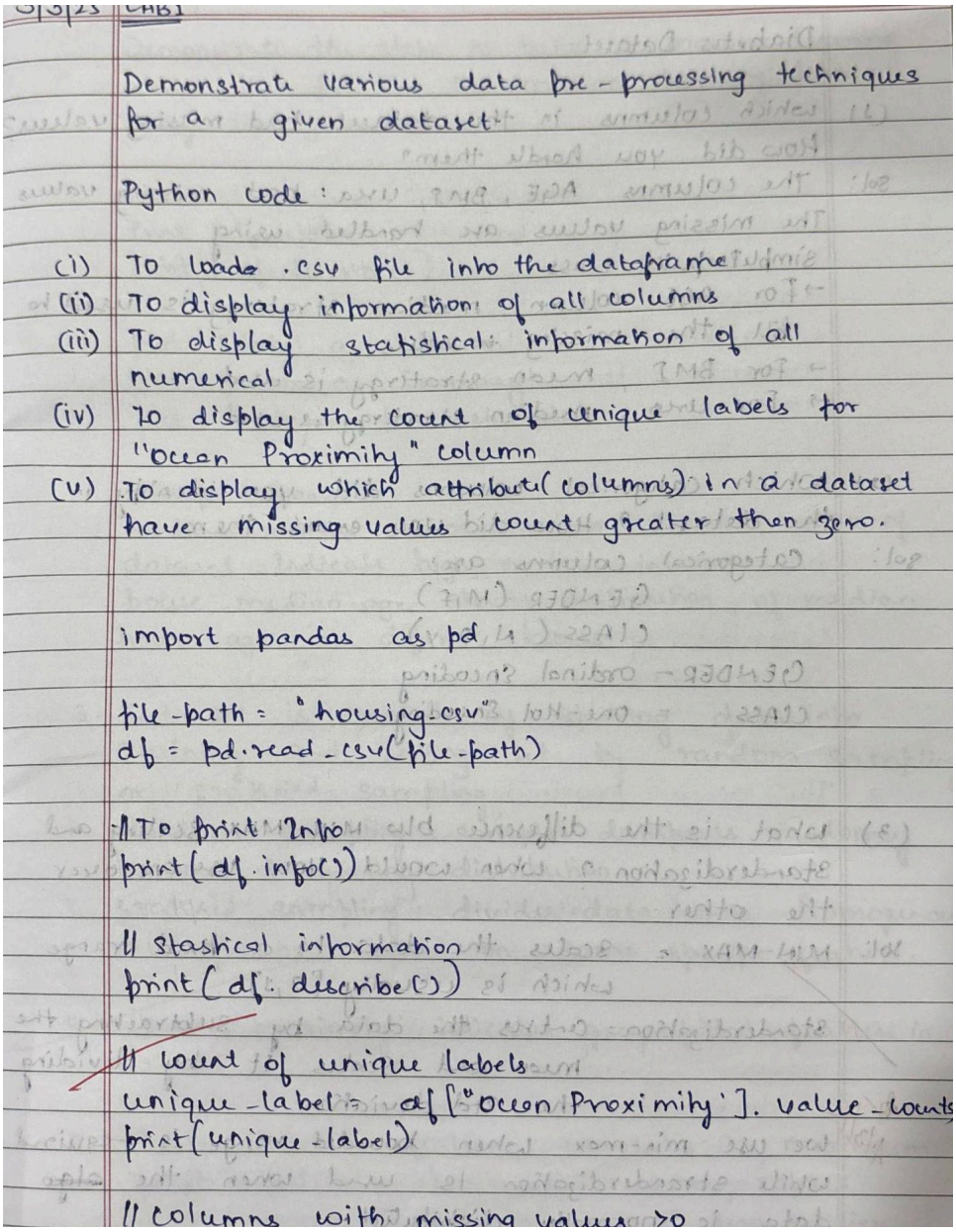
Index

Github Link: https://github.com/NPriyanka167/6thSem-ML-Lab

Program 1

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

Screenshot:

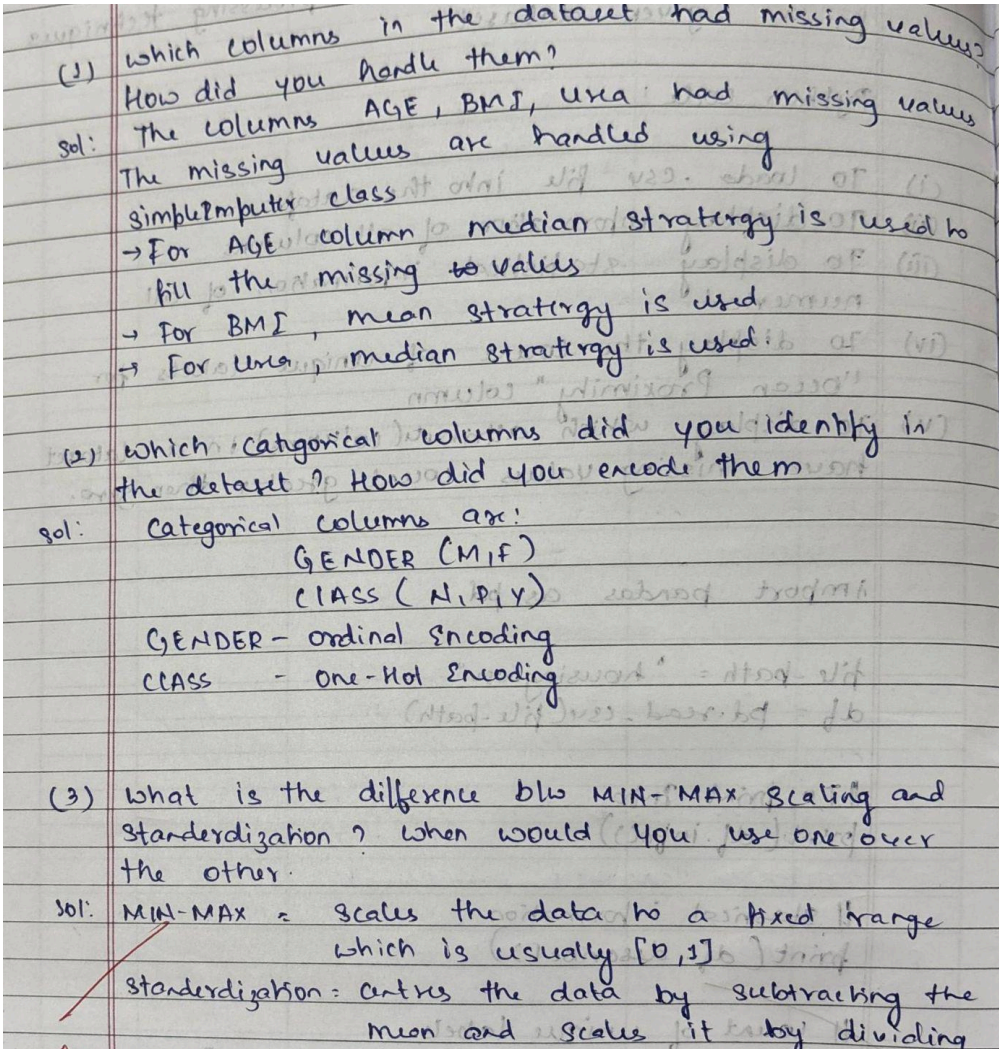


Demonstrate various data pre-processing techniques for a given dataset

Python code:

(i) To loads .csv file into the dataframe
(ii) To display information of all columns
(iii) To display statistical information of all numerical
(iv) To display the count of unique labels for "Ocean Proximity" column
(v) To display which attribute(columns) in a dataset have missing values count greater then zero.

```
import pandas as pd

file_path = "housing.csv"
df = pd.read_csv(file_path)

// To print info
print(df.info())

// stastical information
print(df.describe())

// count of unique labels
unique_label = df["Ocean Proximity"].value_counts
print(unique_label)

// columns with missing values >0
```



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")
```

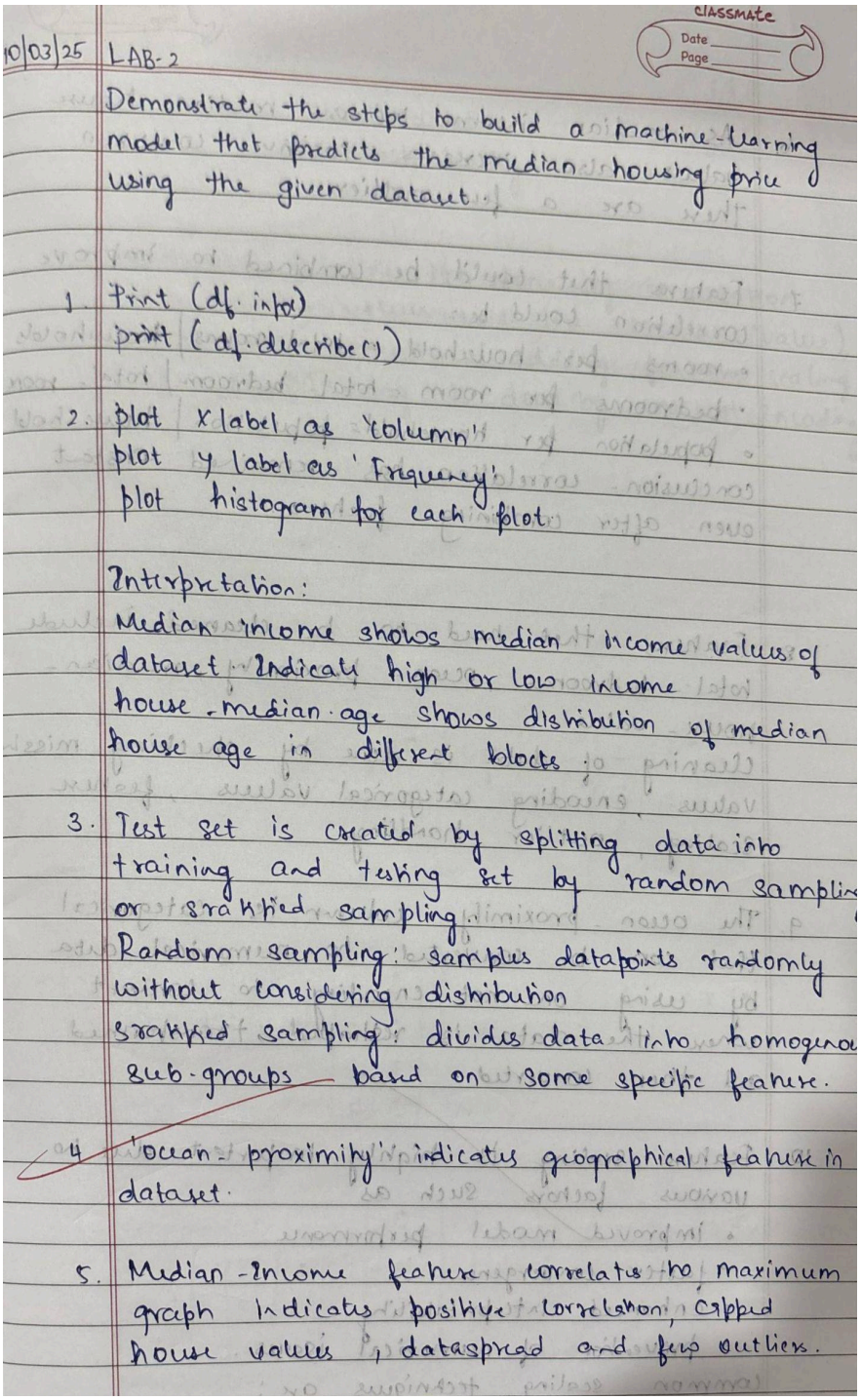
```
plt.subplot(2, 1, 2)
reliance_data['Daily Return'].plot(title="Reliance Industries - Daily Returns", color='orange')
plt.tight_layout()
plt.show()
reliance_data.to_csv('reliance_stock_data.csv')

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
```

Program 2

Demonstrate various data pre-processing techniques for a given dataset

Screenshot:



10/03/25 LAB-2

Demonstrate the steps to build a machine-learning model that predicts the median housing price using the given dataset.

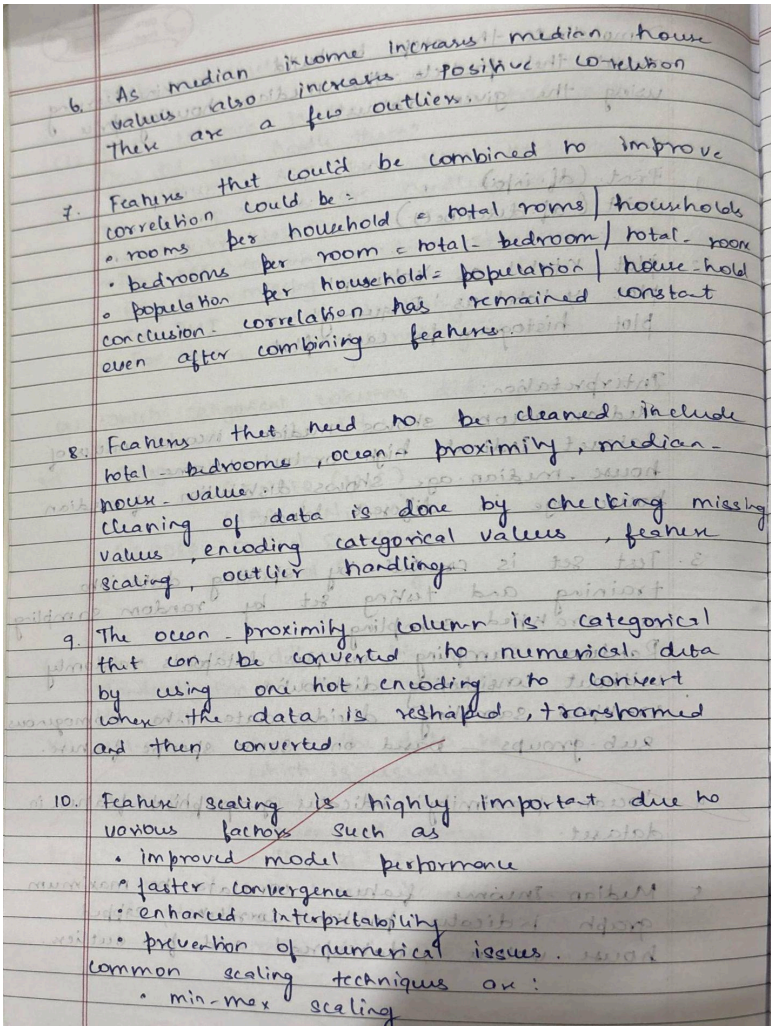
1. Print (df.info)
   print (df.describe())
2. plot x label as 'column'
   plot y label as 'Frequency'
   plot histogram for each plot

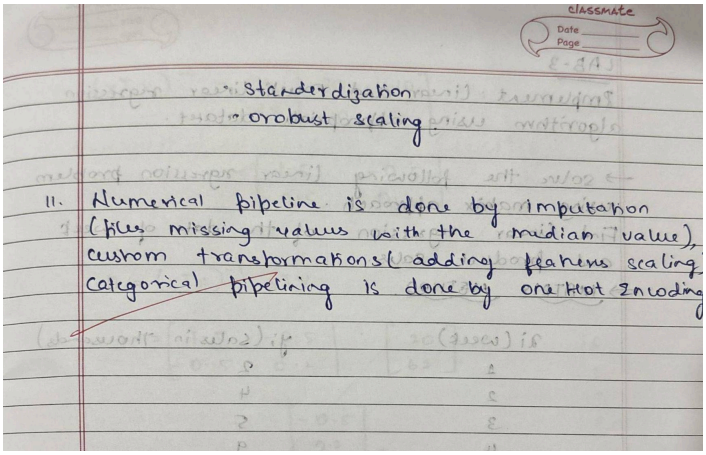
Interpretation:
Median income shows median income values of dataset. Indicate high or low income
house_median_age shows distribution of median house age in different blocks.

3. Test set is created by splitting data into training and testing set by random sampling or stratified sampling.
   Random sampling: samples datapoints randomly without considering distribution
   stratified sampling: divides data into homogenous sub-groups based on some specific feature.

4. 'ocean_proximity' indicates geographical feature in dataset.

5. Median_income feature correlates to maximum graph indicates positive correlation, capped house values, dataspread and few outliers.





Code:

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

from google.colab import files
```

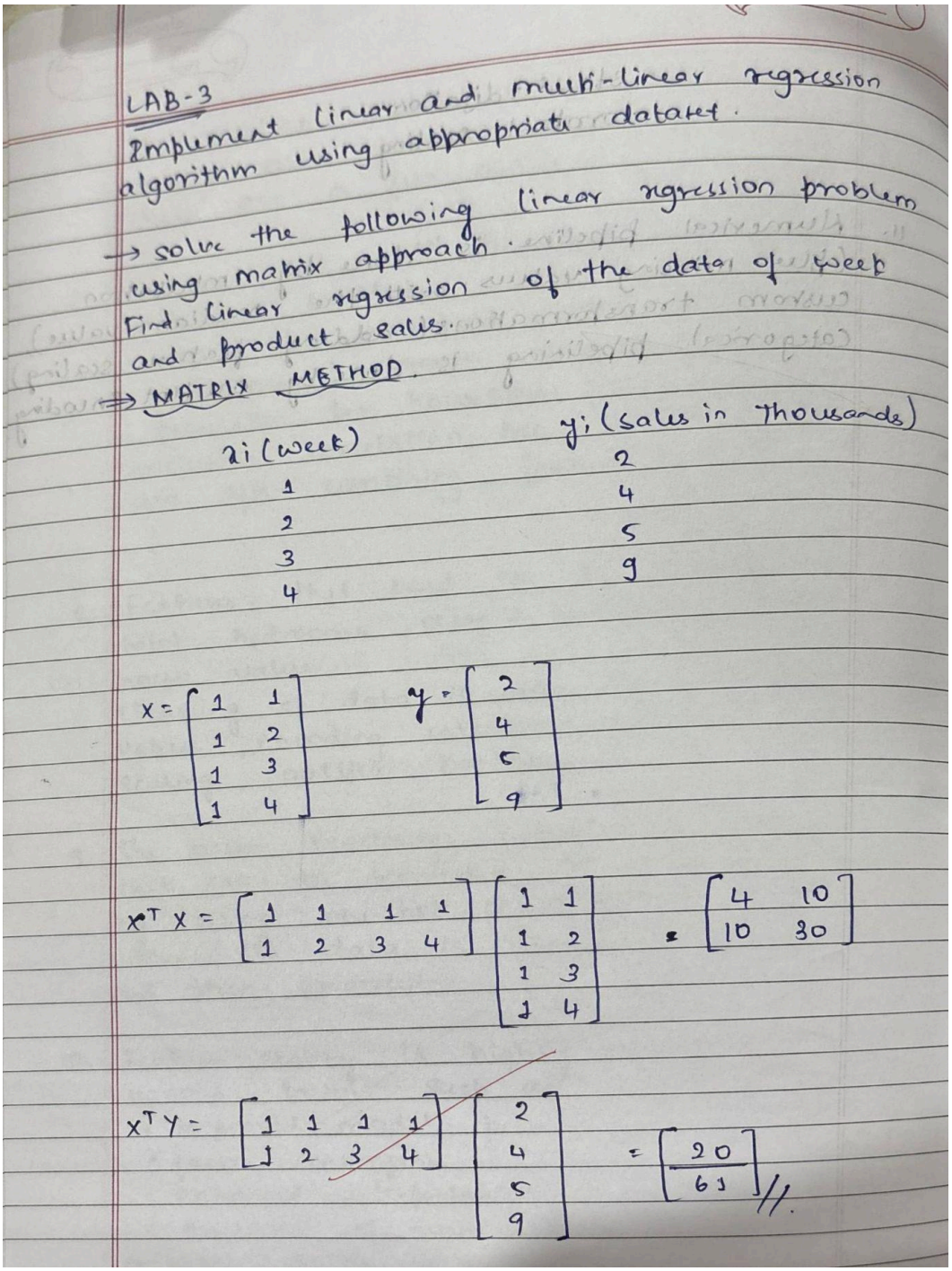
```
adult_income=files.upload()

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, inplace=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:



$$(X^T X)^{-1} = \frac{1}{20}\begin{bmatrix} 30 & -10 \\ -10 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} 1.5 & -0.5 \\ -0.5 & 0.2 \end{bmatrix}$$

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

$$= \begin{bmatrix} 1.5 & -0.5 \\ -0.5 & 0.2 \end{bmatrix} \cdot \begin{bmatrix} 20 \\ 61 \end{bmatrix}$$

$$= \begin{bmatrix} -0.5 \\ 2.2 \end{bmatrix} //$$

Equation of regression line:

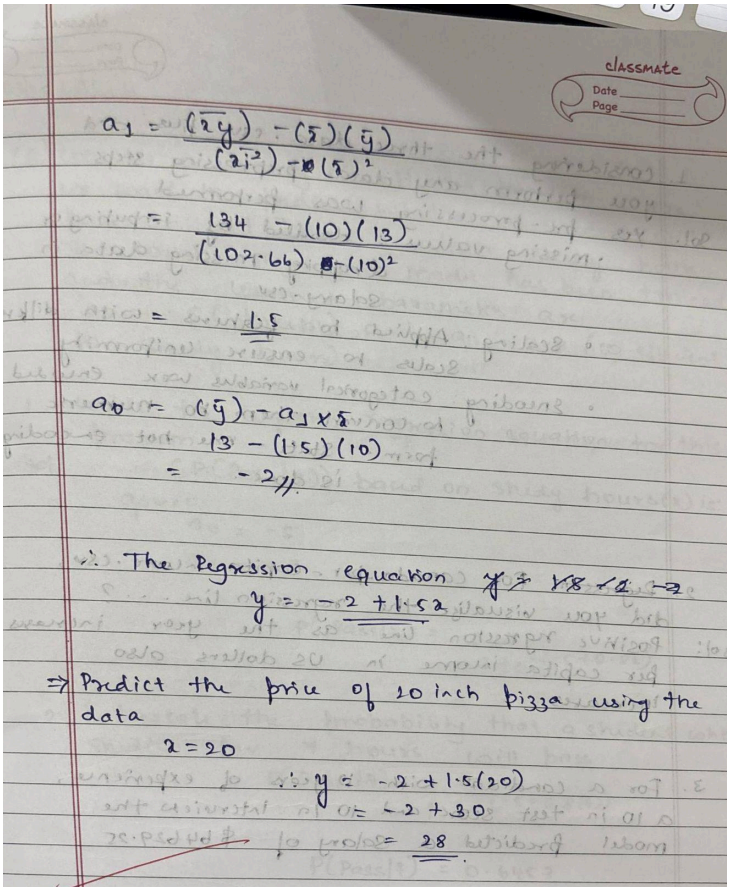
$$y = B_0 + B_1 x$$

$$y = -0.5 + 2.2x$$

⇒ NORMAL METHOD

Diameter (x)	Price (Y)
8	10
10	13
12	16

x_i	y_i	$x_i x_i$	$x_i y_i$
8	10	64	80
10	13	100	130
12	16	144	192
Sum = 30	Sum = 39	Sum = 308	Sum = 402
Avg(x_i) = 30/3 = 10	Avg(y_i) = 39/3 = 13	Avg($x_i x_i$) = 308/3 = 102.66	Avg($x_i y_i$) = 402/3 = 134



$$a_1 = \frac{(\bar{xy}) - (\bar{x})(\bar{y})}{(\bar{x^2}) - (\bar{x})^2}$$

$$= \frac{134 - (10)(13)}{(102.66) - (10)^2}$$

$$= 1.5$$

$$a_0 = (\bar{y}) - a_1 \times \bar{x}$$

$$13 - (1.5)(10)$$

$$= -2 //.$$

∴ The Regression equation

$$y = -2 + 1.5x$$

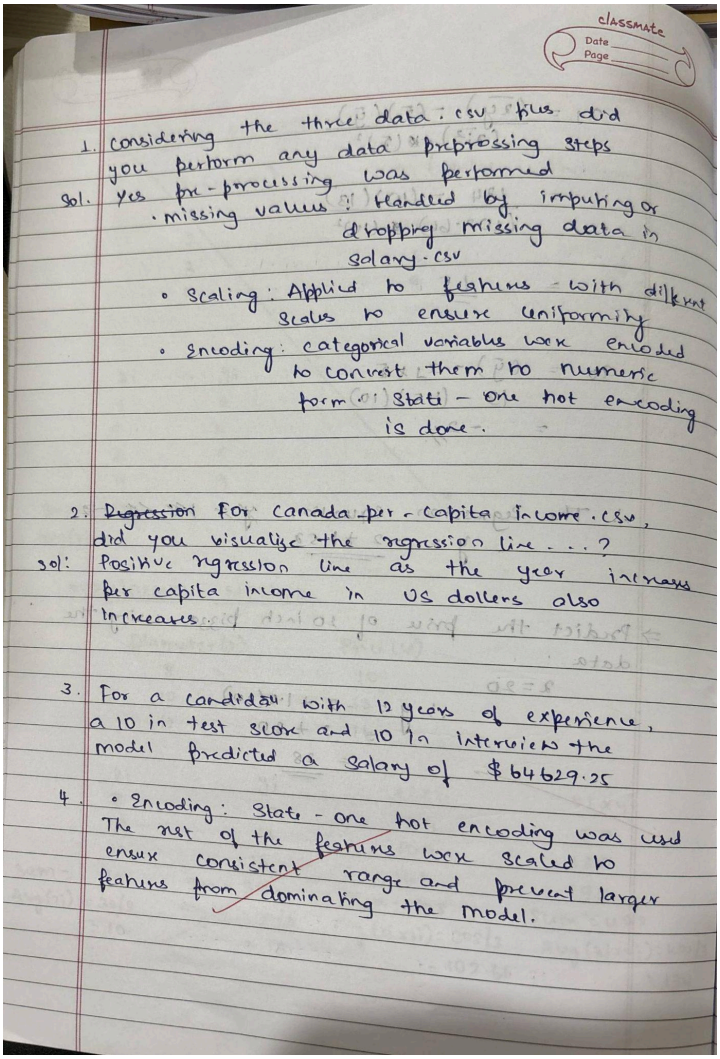
⇒ Predict the price of 20 inch pizza using the data

$x = 20$

$\therefore y = -2 + 1.5(20)$

$= -2 + 30$

$= 28$.



1. considering the three data.csv files did you perform any data preprocessing steps

sol. Yes pre-processing was performed
- missing values: Handled by imputing or dropping missing data in salary.csv
- Scaling: Applied to features with different scales to ensure uniformity
- Encoding: categorical variables were encoded to convert them to numeric form. Stati - one hot encoding is done.

2. ~~Regression~~ For canada_per_capita_income.csv, did you visualise the regression line...?

sol: Positive regression line as the year increases per capita income in US dollers also increases.

3. For a candidate with 12 years of experience, a 10 in test score and 10 in interviews the model predicted a salary of $64629.25

4. • Encoding: State - one hot encoding was used The rest of the features were scaled to ensure consistent range and prevent larger features from dominating the model.

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()
```

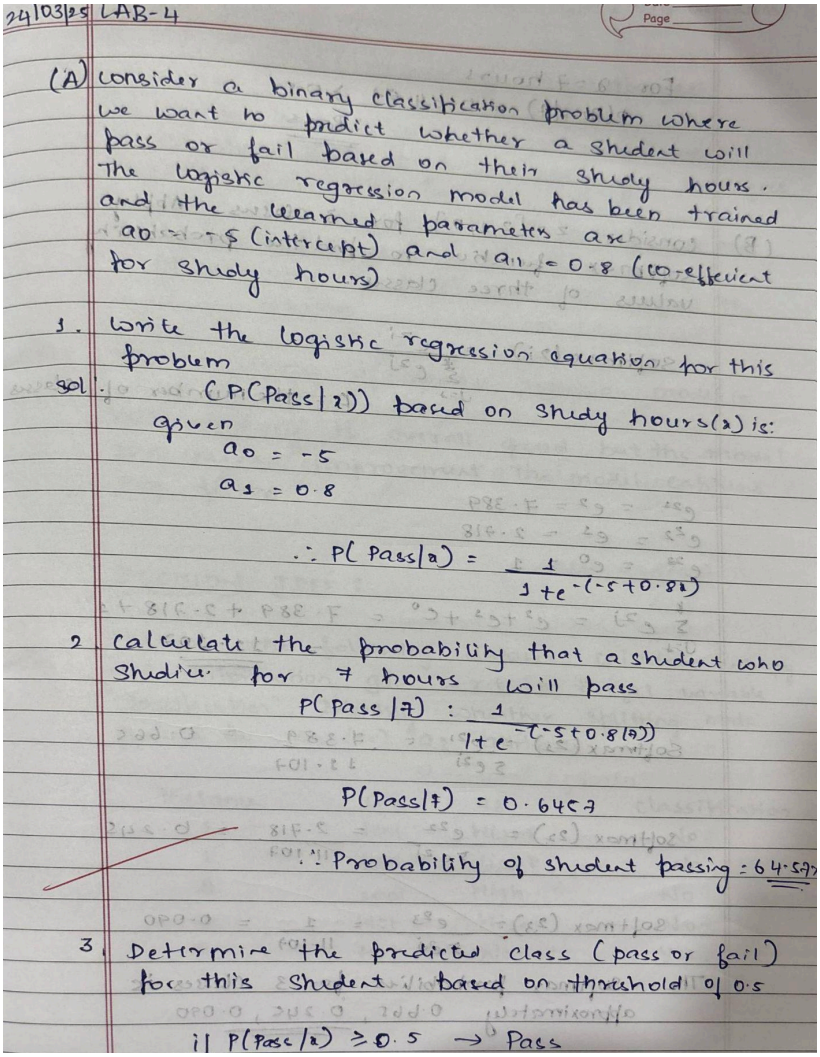
```
df2=pd.read_csv("salary.csv")
df2.head()


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:



24/03/25 LAB-4

(A) 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 $a_0 = -5$ (intercept) and $a_1 = 0.8$ (co-efficient for study hours)

1. Write the logistic regression equation for this problem

Sol: $(P(Pass|x))$ based on study hours(x) is:

given

$a_0 = -5$

$a_1 = 0.8$

$\therefore P(Pass|x) = \frac{1}{1+e^{-(-5+0.8x)}}$

2. Calculate the probability that a student who studies for 7 hours will pass

$P(Pass|7) = \frac{1}{1+e^{-(-5+0.8(7))}}$

$P(Pass|7) = 0.6457$

$\therefore$ Probability of student passing : 64.57%

3. Determine the predicted class (pass or fail) for this student based on threshold of 0.5

if $P(Pass|x) \geq 0.5 \rightarrow$ Pass

For $x = 7$ hours

$P(\text{Pass}|7) \approx 0.6457 \geq 0.5$

$\therefore$ Pass

(B) Consider $z = [2, 1, 0]$ for 3 classes. Apply softmax function to find the probability values of three classes.

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

k = the number of class

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

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

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

$\sum_{j=1}^{k} e^{z_j} = e^2 + e^1 + e^0 = 7.389 + 2.718 + 1 = 11.107$

$$\text{Softmax}(z_1) = \frac{e^{z_1}}{\sum e^{z_j}} = \frac{7.389}{11.107} = 0.665$$

$$\text{Softmax}(z_2) = \frac{e^{z_2}}{\sum e^{z_j}} = \frac{2.718}{11.107} = 0.245$$

$$\text{Softmax}(z_3) = \frac{e^{z_3}}{\sum e^{z_j}} = \frac{1}{11.107} = 0.090$$

The softmax probabilities for 3 classes is approximately 0.665, 0.245, 0.090

(8) After Building the regression model write the answers for the following question

1. The key variables impacting the employee retention are:
   - Satisfaction level - lower satisfaction increases attrition
   - time spent in company - employees with 5+ years tend to leave
   - salary - low salaries lead to higher turnover.
   - number of projects and average monthly hours - very high (or) low values affect the retention.

2. The accuracy of logistic regression model is 78.40%. The accuracy is overall good but the model still needs improvement the model captures key factors.

Code:

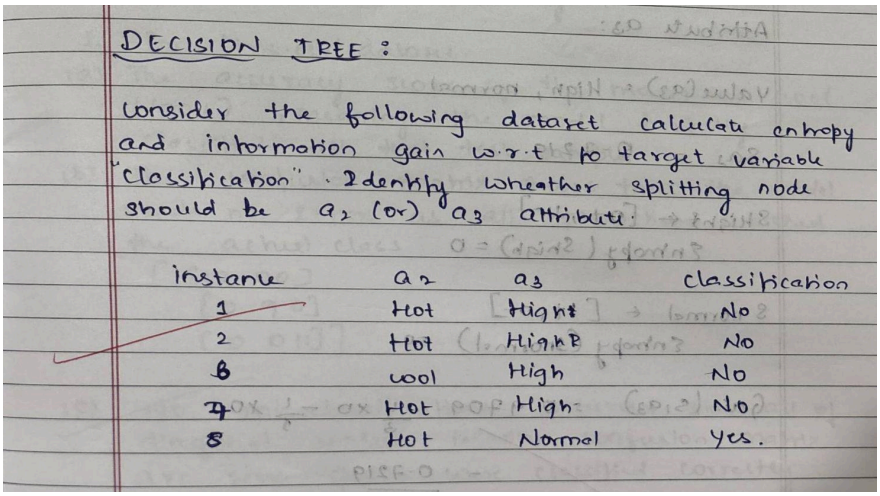
```
from google.colab import files
hr=files.upload()

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 4

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

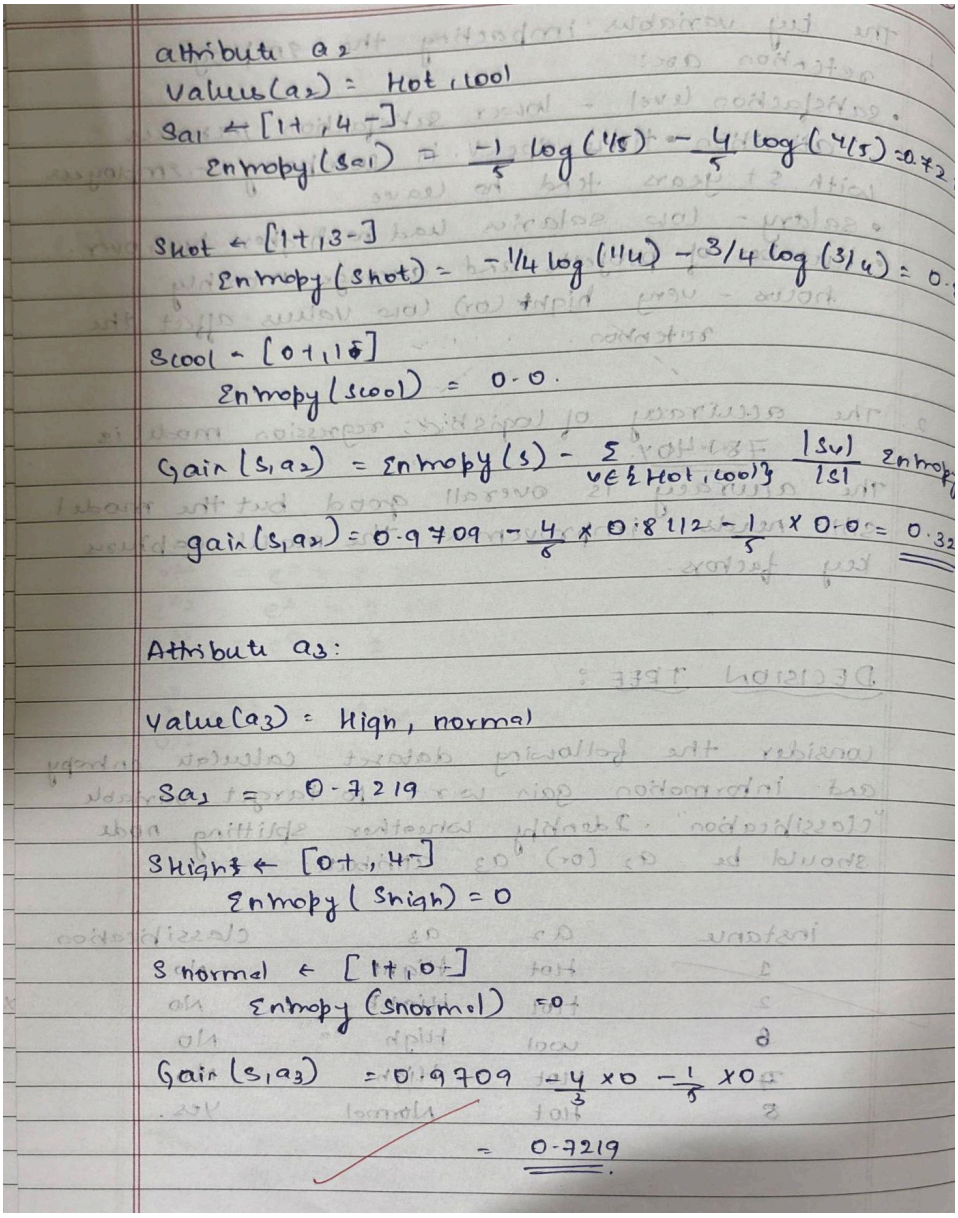
Screenshot:

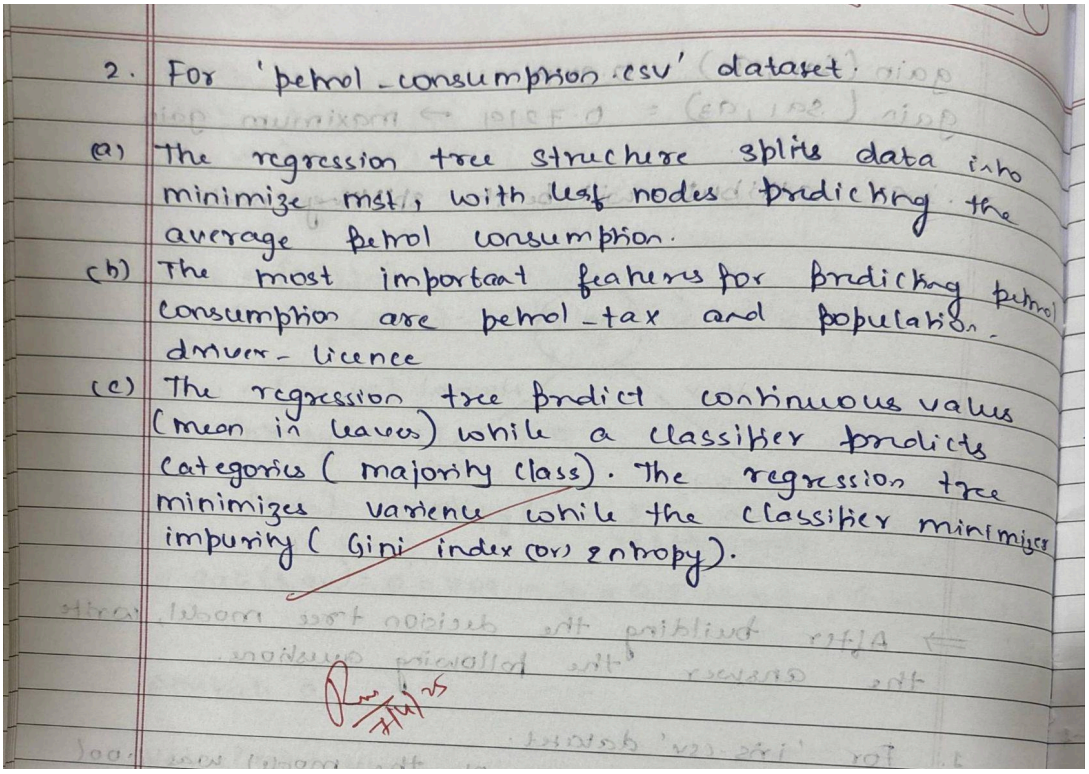


DECISION TREE :

Consider the following dataset calculate entropy and information gain w.r.t to target variable "classification". Identify wheather splitting node should be a_2 (or) a_3 attribute.

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.


attribute a2
Value(a2) = Hot, cool
Sa1 ← [1+, 4-]
Entropy(Sa1) = -1/5 log(1/5) - 4/5 log(4/5) = 0.72
Shot ← [1+, 3-]
Entropy(Shot) = -1/4 log(1/4) - 3/4 log(3/4) = 0.
Scool ← [0+, 1-]
Entropy(Scool) = 0.0.
Gain(S, a2) = Entropy(S) - Σ v∈{Hot, cool} |Sv|/|S| Entropy
gain(S, a2) = 0.9709 - 4/5 × 0.8112 - 1/5 × 0.0 = 0.32
Attribute a3:
Value(a3) = High, normal
Sa3 = 0.7219
SHigh ← [0+, 4-]
Entropy(Shigh) = 0
Snormal ← [1+, 0-]
Entropy(Snormal) = 0
Gain(S, a3) = 0.9709 - 4/5 × 0 - 1/5 × 0
= 0.7219



2. For 'petrol_consumption.csv' dataset.

(a) The regression tree structure splits data into minimize mse, with leaf nodes predicting the average petrol consumption.

(b) The most important features for predicting petrol consumption are petrol_tax and population. driver_licence

(c) The regression tree predict continuous values (mean in leaves) while a classifier predicts categories (majority class). The regression tree minimizes variance while the classifier minimizes impurity (Gini index or entropy).

Code:

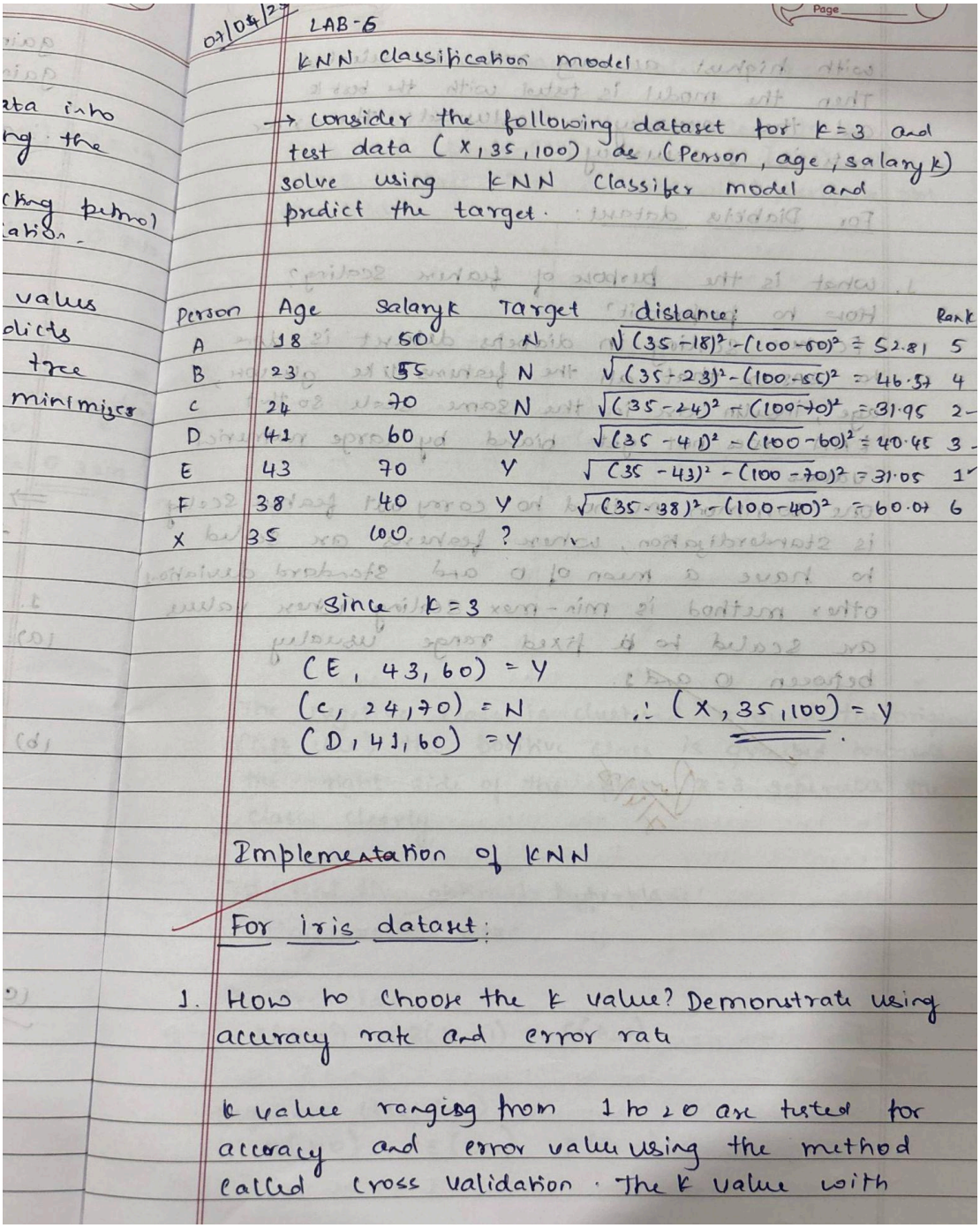
```
from google.colab import files
iris=files.upload()
df1=pd.read_csv("iris.csv")
df1.head()

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 5

Build KNN Classification model for a given dataset.

Screenshot:



07/04/24 LAB-6

KNN classification model

→ Consider the following dataset for k=3 and test data (x, 35, 100) de (Person, age, salary k) solve using KNN classifier model and predict the target.

Person	Age	Salary k	Target	distance	Rank
A	18	50	N	$\sqrt{(35-18)^2-(100-50)^2} \approx 52.81$	5
B	23	55	N	$\sqrt{(35-23)^2-(100-55)^2} = 46.57$	4
C	24	70	N	$\sqrt{(35-24)^2-(100-70)^2} = 31.95$	2
D	41	60	Y	$\sqrt{(35-41)^2-(100-60)^2} \approx 40.45$	3
E	43	70	Y	$\sqrt{(35-43)^2-(100-70)^2} = 31.05$	1
F	38	40	Y	$\sqrt{(35-38)^2-(100-40)^2} \approx 60.07$	6
X	35	100	?		

Since k = 3

(E, 43, 60) = Y

(C, 24, 70) = N

(D, 41, 60) = Y

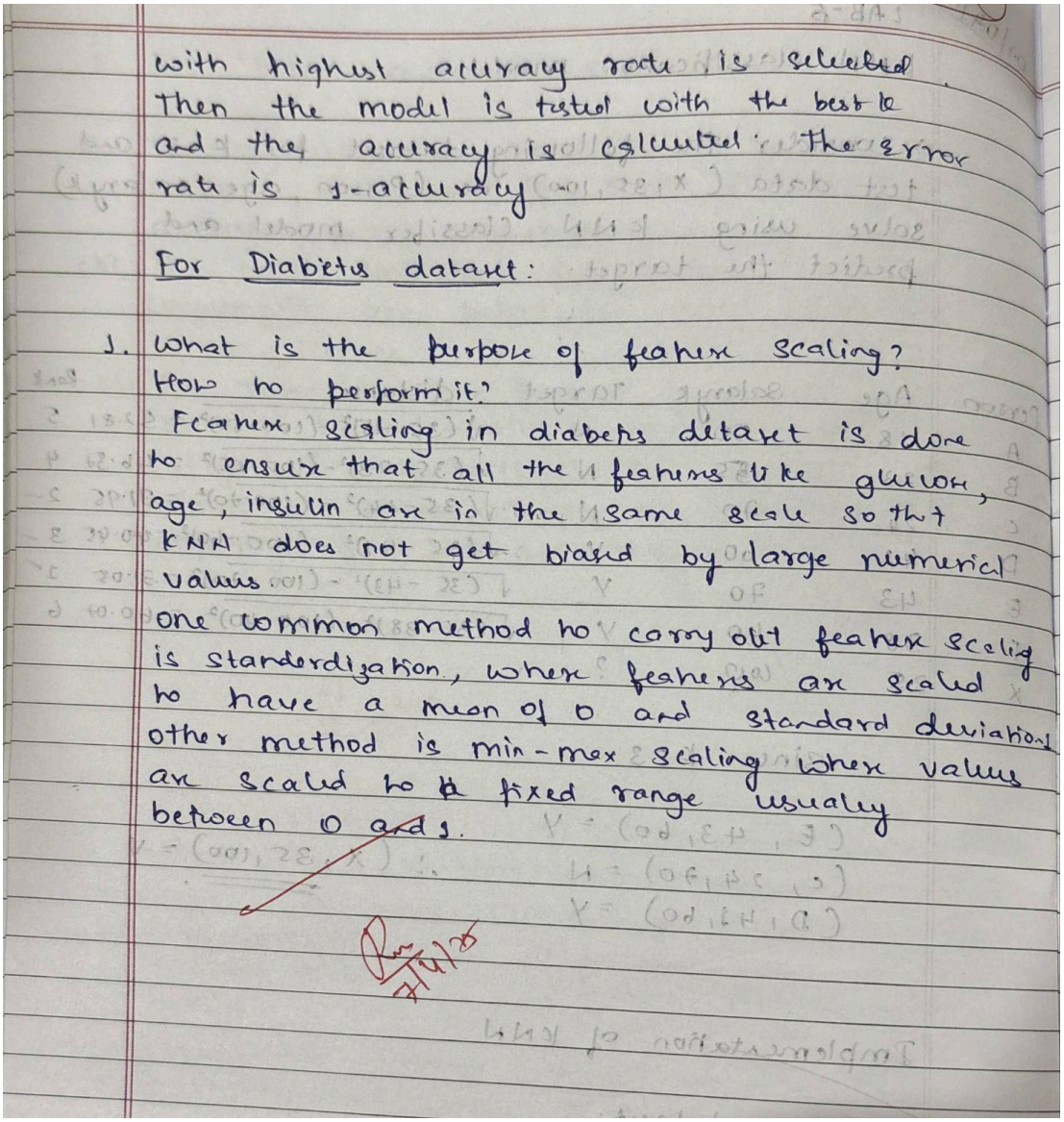
∴ (X, 35, 100) = Y

Implementation of KNN

For iris dataset:

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

K value ranging from 1 to 20 are tested for accuracy and error value using the method called cross validation. The K value with



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}")
```

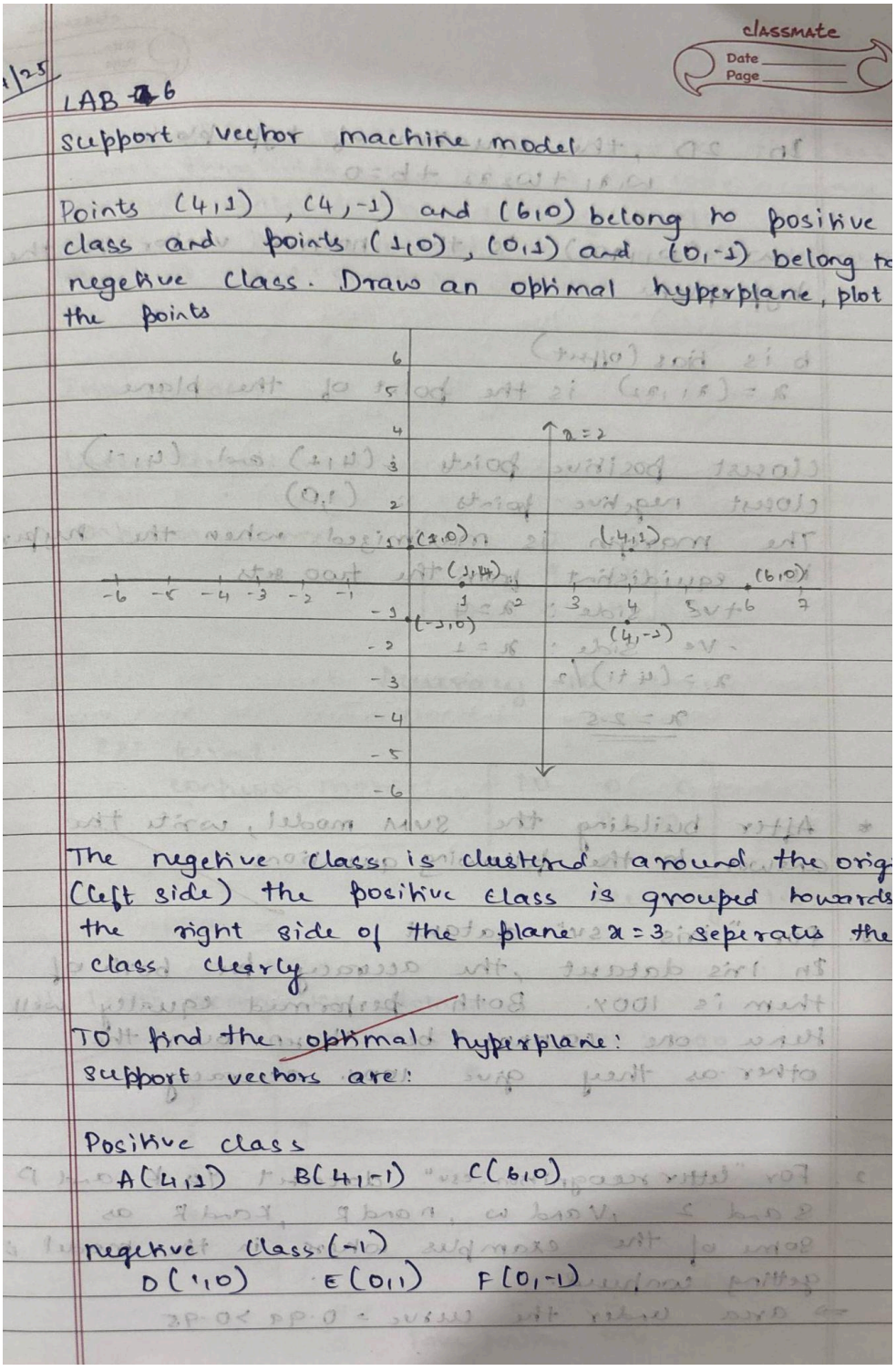
```
    if accuracy >
        best_accuracy:
        best_accuracy = accuracy
        best_k = k
print(f"Best k value: {best_k}")
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Accuracy Score:", accuracy_score(y_test, y_pred))
print("\nConfusion 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()
```

Program

6 Build Support vector machine model for a given dataset

Screenshot:



In 2D, the equation of hyperplane is
$w_1 x_1 + w_2 x_2 + b = 0$

$w = (w_1, w_2)$ is the normal vector to the hyperplane

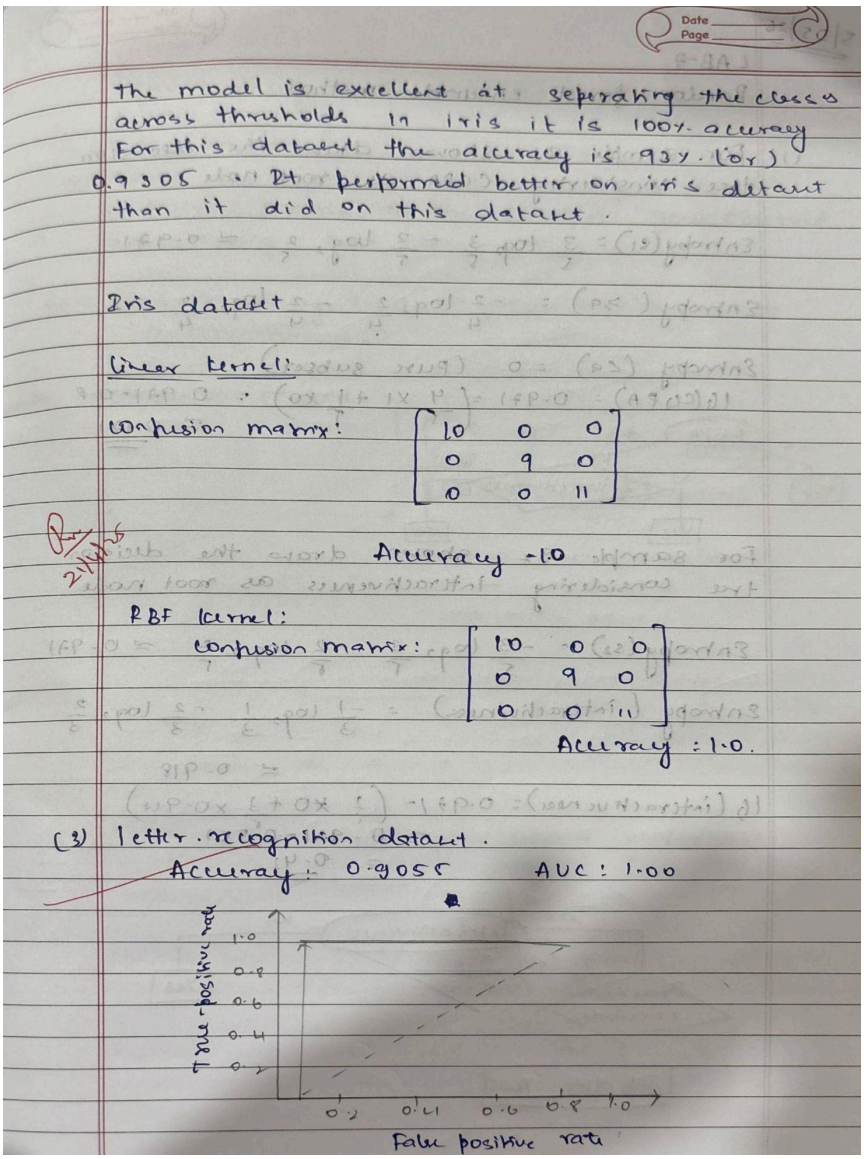
b is bias (offset)
$x = (x_1, x_2)$ is the point of the plane

Closest positive points : (4,2) and (4,-1)
closest negative points : (1,0)
The margin is maximized when the hyperplane is equidistant from the two sets
+ve side : $x = 4$
-ve side : $x = 1$
$x = (4+1)/2$
$x = 2.5$

* After building the SVM model, write the answer to the following questions

1. For "iris.csv" dataset
In iris dataset, the accuracy for both of them is 100%. Both performed equally well. Hence one cannot be chosen over the other as they give 100% accuracy.

2. For "letter recognition.csv" dataset Yes K and P
8 and 2, V and W, N and R, K and R as
some of the examples where in the model is getting confused.
⇒ area under the curve = 0.99 > 0.95



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')
```

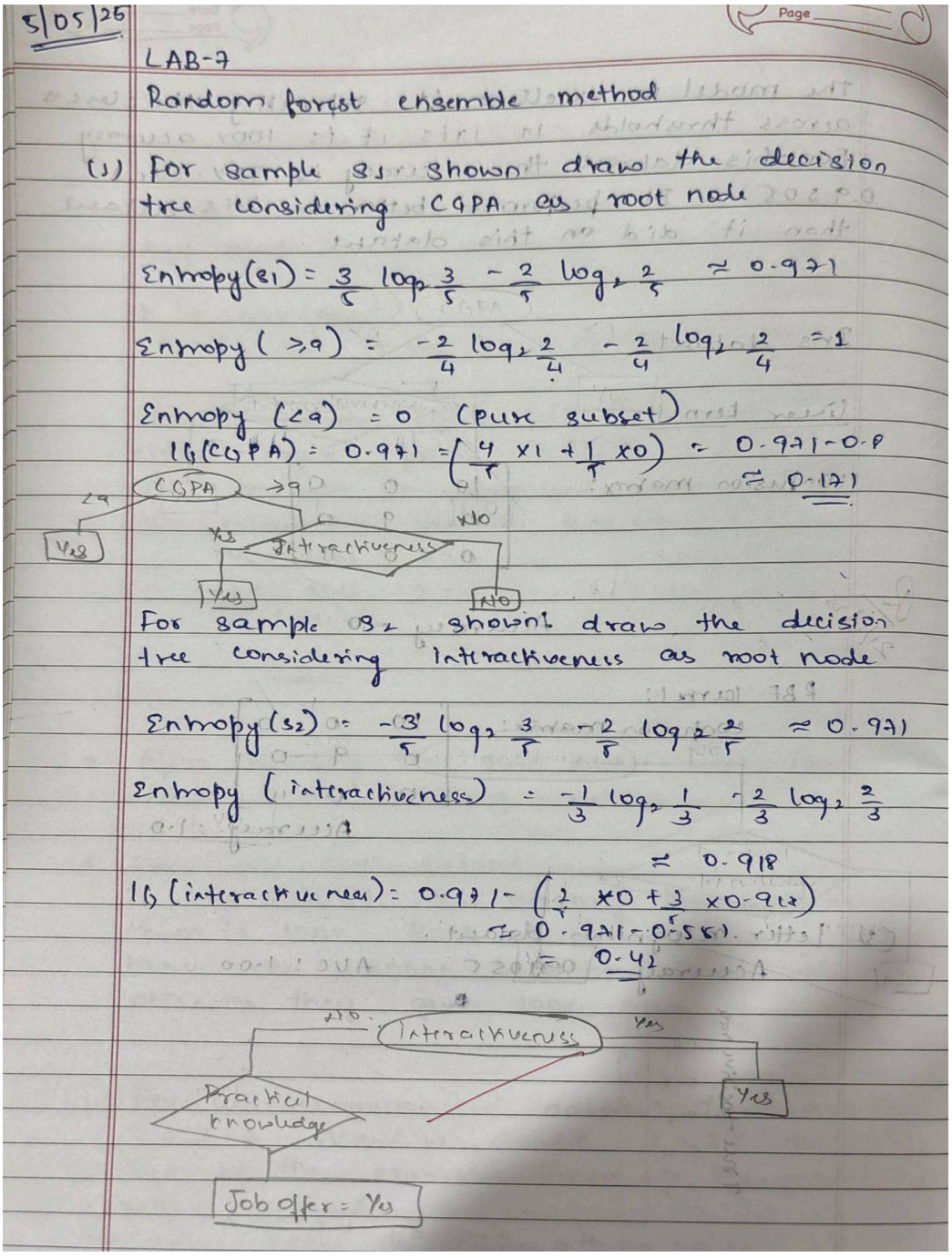
```
plt.xlabel('Predicted')


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()
```

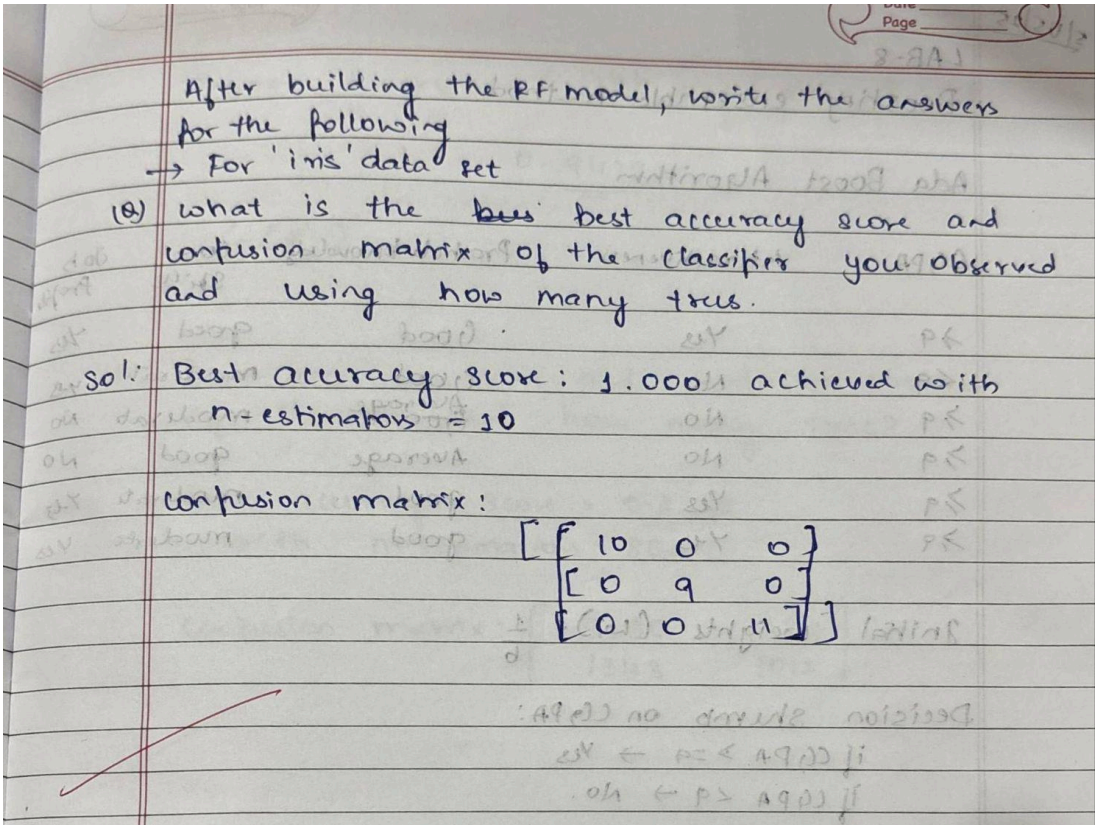
```
print(f"AUC score for class 0: {roc_auc[0]}"
```

Program 7

Implement Random forest ensemble method on a given dataset

Screenshot:





Code:

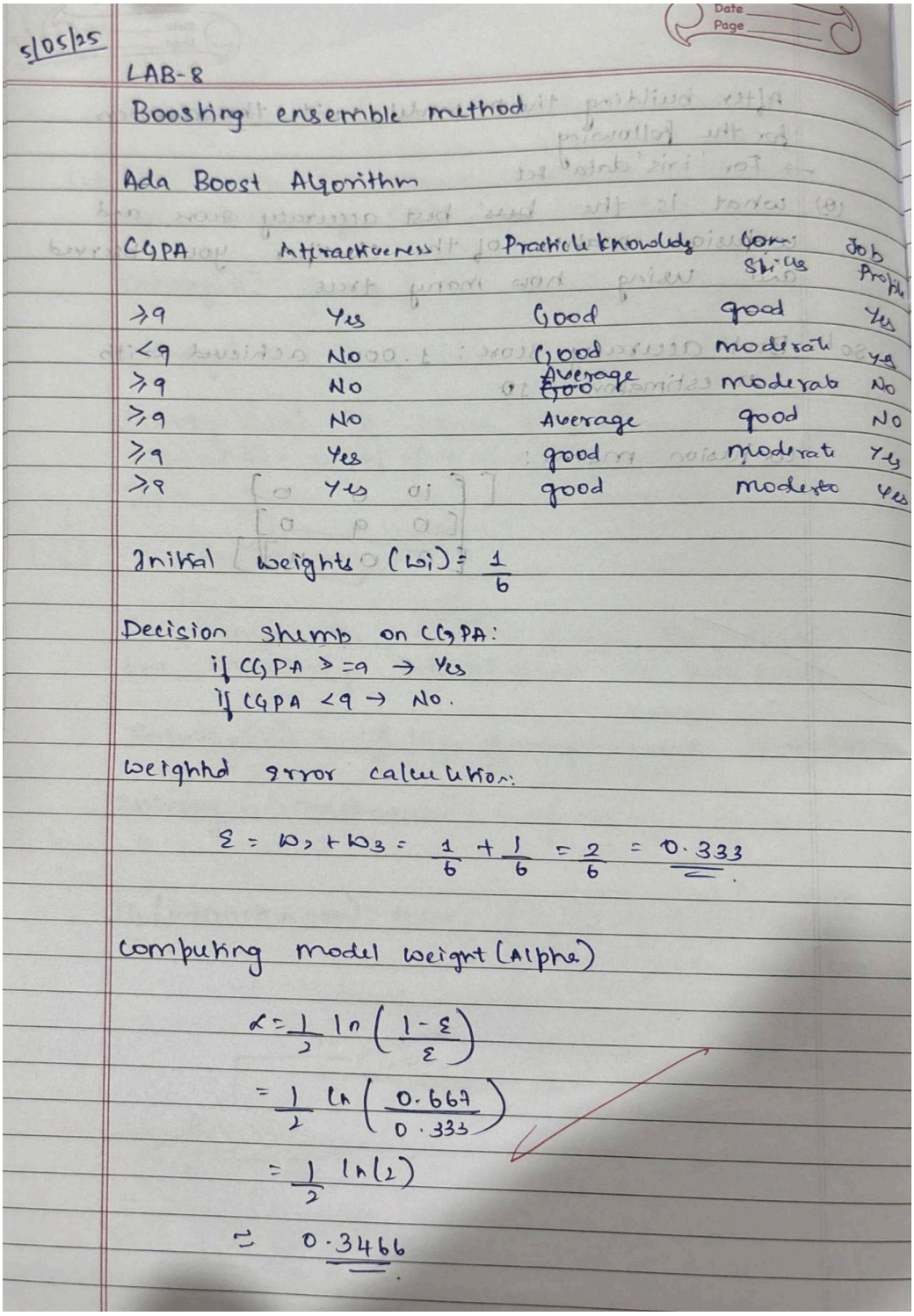
```
from google.colab import files
iris=files.upload()
df1=pd.read_csv("iris (4).csv")
df1.head()
X = df1.drop('species', axis=1)
y = df1['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
rf_classifier = RandomForestClassifier(random_state=0)
rf_classifier.fit(X_train, 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 8

Implement Boosting ensemble method on a given dataset

Screenshot:



5/05/25

LAB-8

Boosting ensemble method

Ada Boost Algorithm

CGPA	Interactiveness	Practical knowledge	Com Skills	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

Initial weights (w_i) = $\frac{1}{6}$

Decision stump on CGPA:

if CGPA >=9 → Yes

if CGPA <9 → No.

weighted error calculation:

$$\varepsilon = w_2 + w_3 = \frac{1}{6} + \frac{1}{6} = \frac{2}{6} = 0.333$$

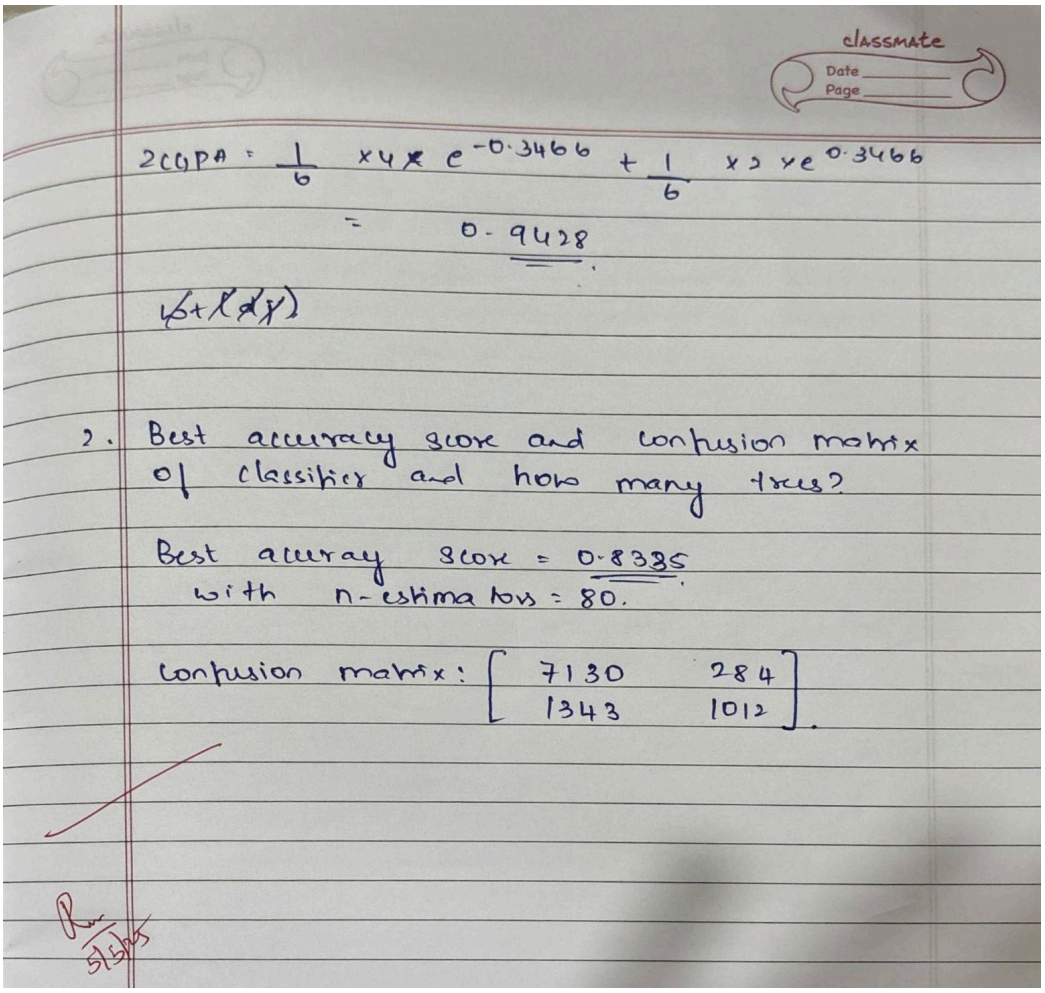
computing model weight (Alpha)

$$\alpha = \frac{1}{2} \ln\left(\frac{1-\varepsilon}{\varepsilon}\right)$$

$$= \frac{1}{2} \ln\left(\frac{0.667}{0.333}\right)$$

$$= \frac{1}{2} \ln(2)$$

$$\simeq 0.3466$$



Code:

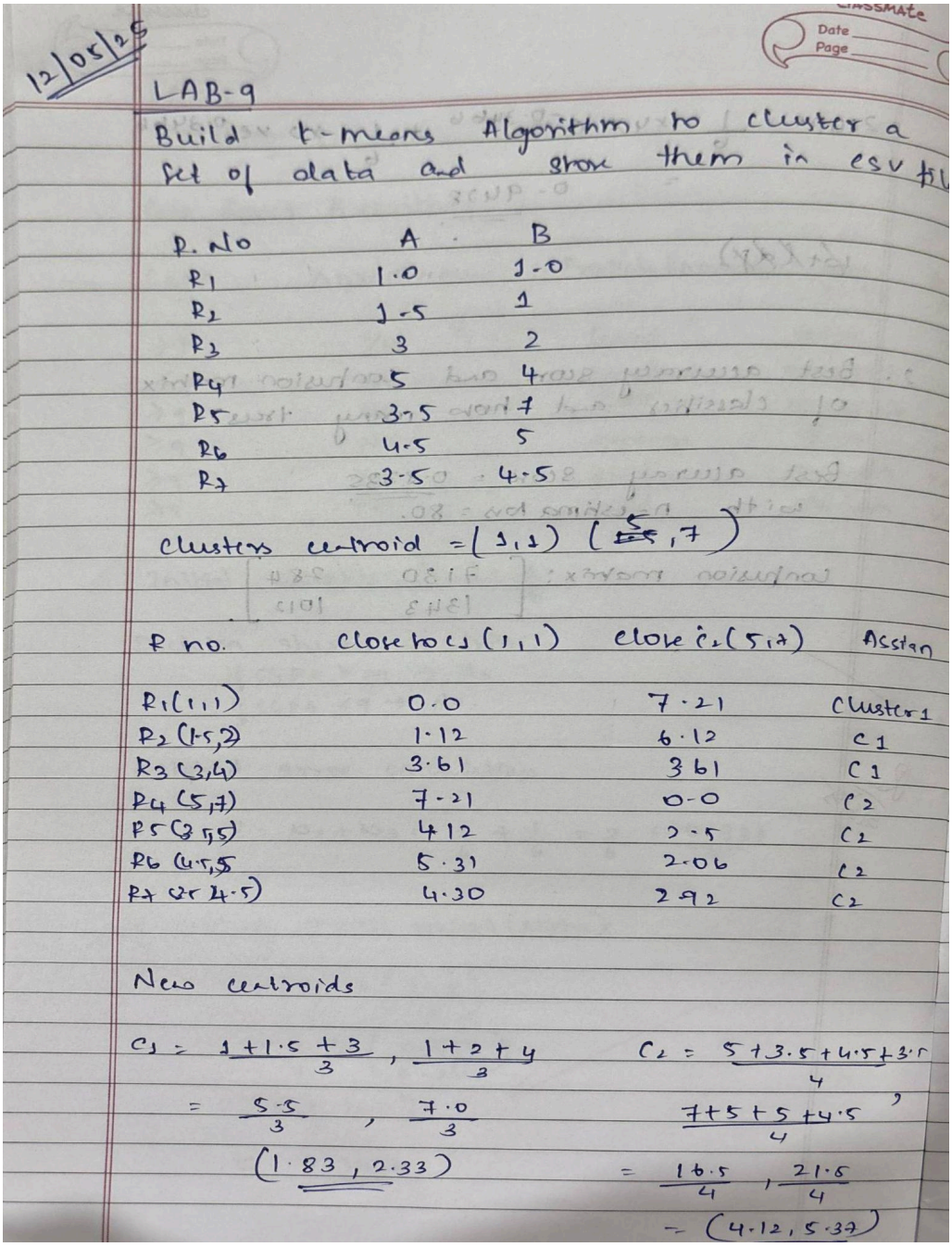
```
from google.colab import files
income=files.upload()
df1=pd.read_csv("income.csv")
df1.head()
X = df1.drop('income_level', axis=1)
```

```
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 9

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

Screenshot:



12/05/25

LAB-9

Build k-means Algorithm to cluster a set of data and store them in csv file

P. No	A	B
P_1	1.0	1.0
P_2	1.5	1
P_3	3	2
P_4	5	4
P_5	3.5	4
P_6	4.5	5
P_7	3.5	4.5

Clusters centroid = (1,1) (5,7)

P no.	close to c_1 (1,1)	close to c_2 (5,7)	Assign
P_1 (1,1)	0.0	7.21	Cluster 1
P_2 (1.5,1)	1.12	6.12	C_1
P_3 (3,4)	3.61	3.61	C_1
P_4 (5,7)	7.21	0.0	C_2
P_5 (3.5,5)	4.12	2.5	C_2
P_6 (4.5,5	5.31	2.06	C_2
P_7 (or 4.5)	4.30	2.92	C_2

New centroids

$C_1 = \left(\frac{1+1.5+3}{3}, \frac{1+2+4}{3}\right) = \left(\frac{5.5}{3}, \frac{7.0}{3}\right) = (1.83, 2.33)$

$C_2 = \left(\frac{5+3.5+4.5+3.5}{4}, \frac{7+5+5+4.5}{4}\right) = \left(\frac{16.5}{4}, \frac{21.5}{4}\right) = (4.12, 5.37)$

2 Iteration-2

Record No.	Close C_1	Close C_2	Assign
R_1	1.57	5.37	C_1
R_2	0.47	4.27	C_1
R_3	2.04	3.77	C_2
R_4	5.64	1.85	C_2
R_5	3.15	0.22	C_2
R_6	3.78	0.53	C_2
R_7	2.74	1.07	C_1

New centroid.

$C_1 = \frac{1+1.5}{2}, \frac{1+2}{2}$

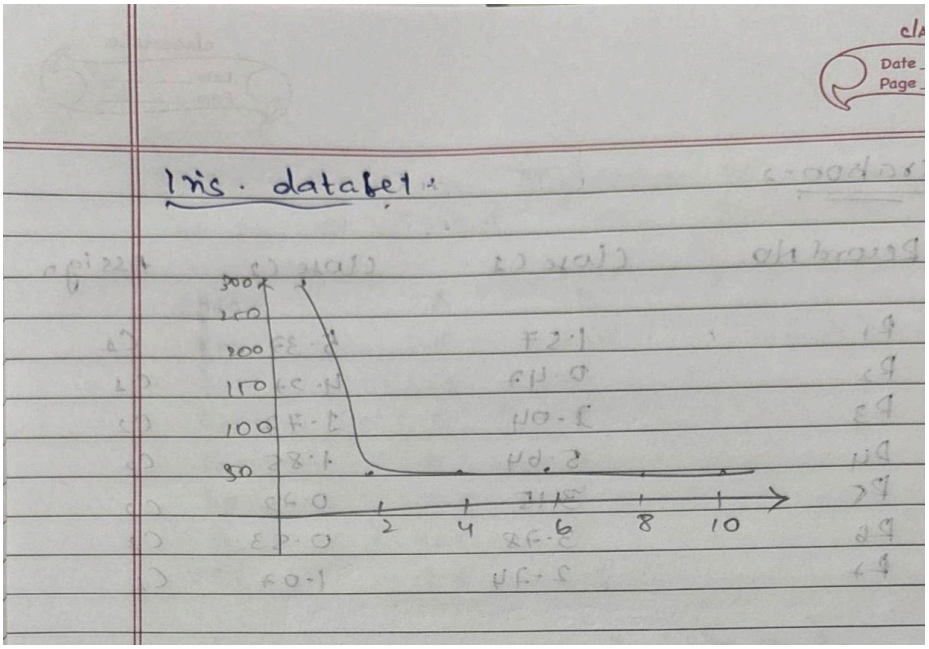
$= (1.25, 1.5)$

$C_2 = \frac{3+5+3.5+4.5+3.5}{5}, \frac{4+7+5+5+4.5}{5}$

$= \frac{19.5}{5}, \frac{25.5}{5}$

$= (3.9, 5.1)$

∴ The two cluster centres after 2 iterations are: C_1 (1.25, 1.5)
C_2 (3.9, 5.1)



Code:

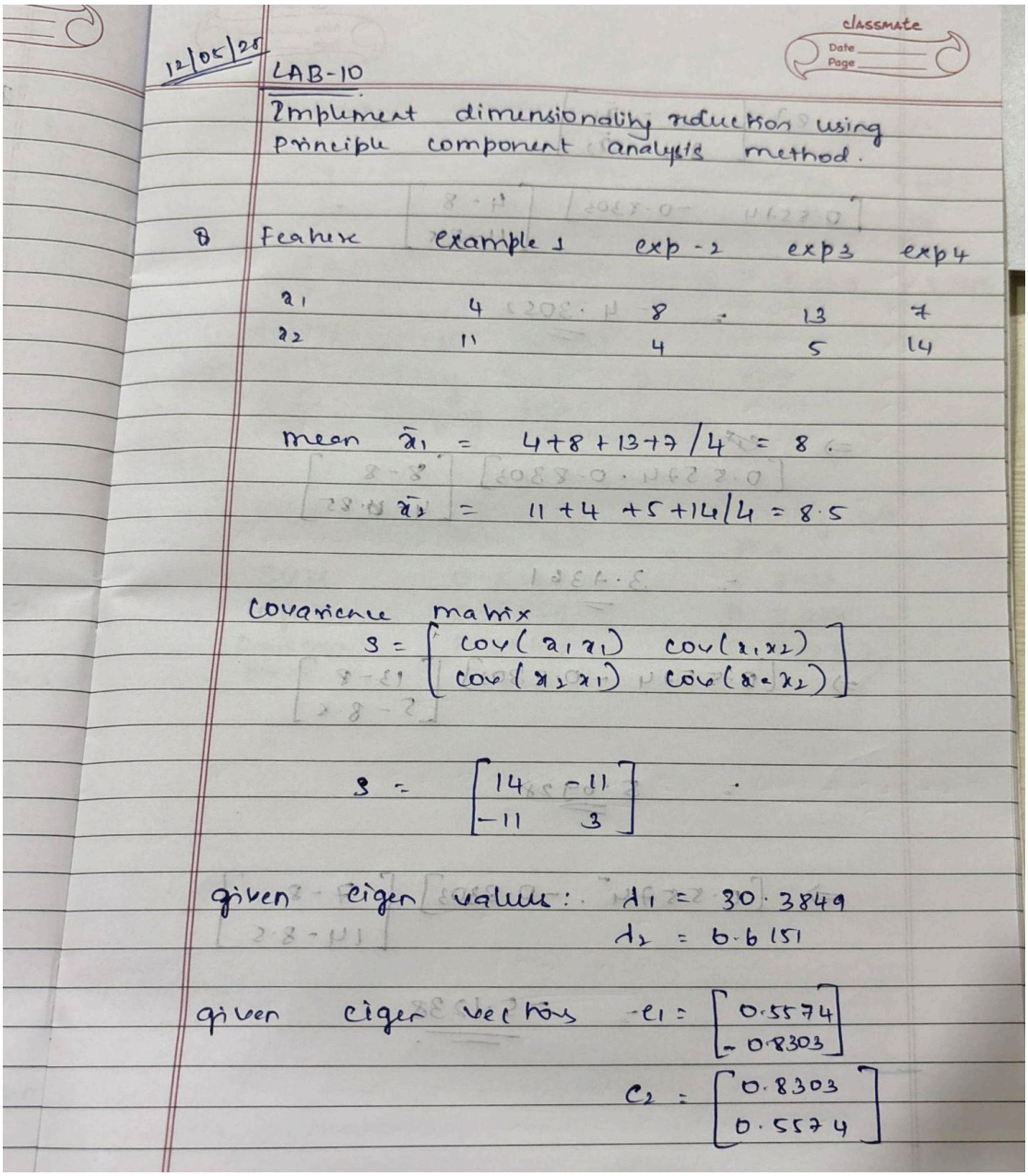
```
from google.colab import files
iris=files.upload()
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from scipy import stats
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import 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 10

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

Screenshot:



12/05/25

LAB-10

Implement dimensionality reduction using principle component analysis method.

S	Feature	example 1	exp-2	exp3	exp4
	x_1	4	8	13	7
	x_2	11	4	5	14

mean $\bar{x}_1 = 4+8+13+7/4 = 8$.

$\bar{x}_2 = 11+4+5+14/4 = 8.5$

Covariance matrix

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

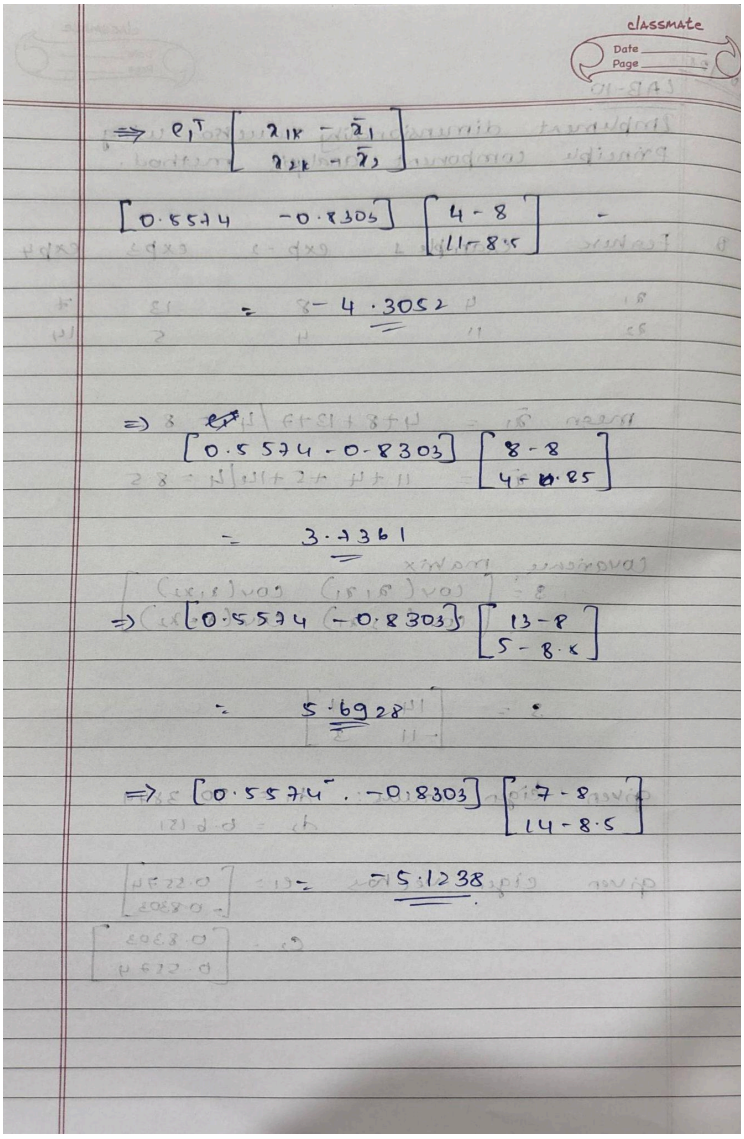
$$S = \begin{bmatrix} 14 & -11 \\ -11 & 3 \end{bmatrix}$$

given eigen values: $\lambda_1 = 30.3849$

$\lambda_2 = 6.6151$

given eigen vectors $e_1 = \begin{bmatrix} 0.5574 \\ -0.8303 \end{bmatrix}$

$e_2 = \begin{bmatrix} 0.8303 \\ 0.5574 \end{bmatrix}$



$$\Rightarrow e_1^T \begin{bmatrix} x_{1k} - \bar{x}_1 \\ x_{2k} - \bar{x}_2 \end{bmatrix}$$

$$\begin{bmatrix} 0.5574 & -0.8305 \end{bmatrix} \begin{bmatrix} 4 - 8 \\ 11 - 8.5 \end{bmatrix}$$

$$= -4.3052$$

$$\Rightarrow \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} 8 - 8 \\ 4 - 8.85 \end{bmatrix}$$

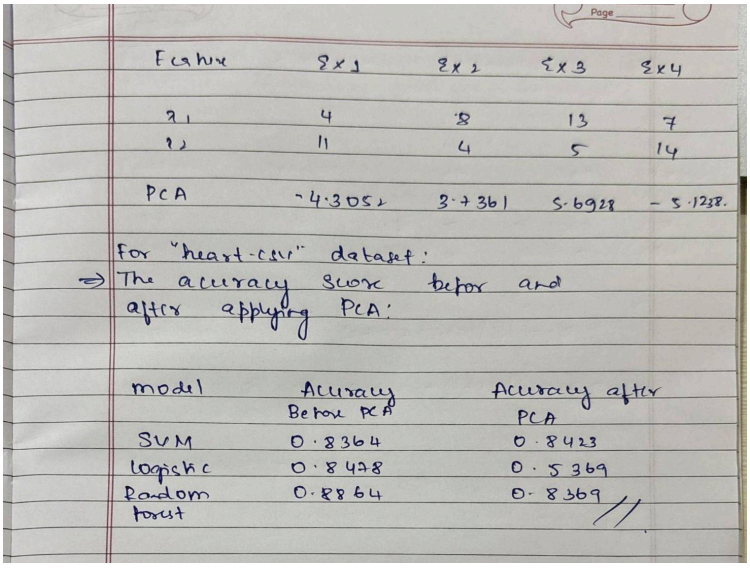
$$= 3.7361$$

$$\Rightarrow \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} 13 - 8 \\ 5 - 8.5 \end{bmatrix}$$

$$= 5.6928$$

$$\Rightarrow \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} 7 - 8 \\ 14 - 8.5 \end{bmatrix}$$

$$= -5.1238$$



Feature	8×1	8×2	8×3	8×4
x_1	4	8	13	7
x_2	11	4	5	14
PCA	-4.3052	3.7361	5.6928	-5.1238

For "heart.csv" dataset:

⇒ The accuracy score before and after applying PCA:

model	Accuracy Before PCA	Accuracy after PCA
SVM	0.8364	0.8423
logistic	0.8478	0.5369
Random forest	0.8864	0.8369

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]}")
```