VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Machine Learning (23CS6PCMAL)

Submitted by

Yashraj Sinha (1BM22CS335)

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 **Yashraj Sinha (1BM22CS335),** who is 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 KM Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

Index

Github Link:

https://github.com/sweet-er-est/ML-Lab

Program 1

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

Code:

```
import pandas as pd

# Method-1: Initializing values directly into DataFrame

data_method1 = {'USN': ['1JS17CS001', '1JS17CS002', '1JS17CS003',

'1JS17CS004', '1JS17CS005'],

'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],

'Marks': [90, 85, 92, 78, 88]}

df_method1 = pd.DataFrame(data_method1)

print("Method-1:")

print(df_method1)

print("-" * 20)

# Method-2: Importing datasets from sklearn.datasets

from sklearn.datasets import load_diabetes

diabetes_data = load_diabetes()

df_method2 = pd.DataFrame(data=diabetes_data.data,

columns=diabetes_data.feature_names)

df_method2['target'] = diabetes_data.target

print("Method-2:")

print(df_method2.head())

print("-" * 20)

# Method-3: Importing datasets from a specific .csv file

try:
```

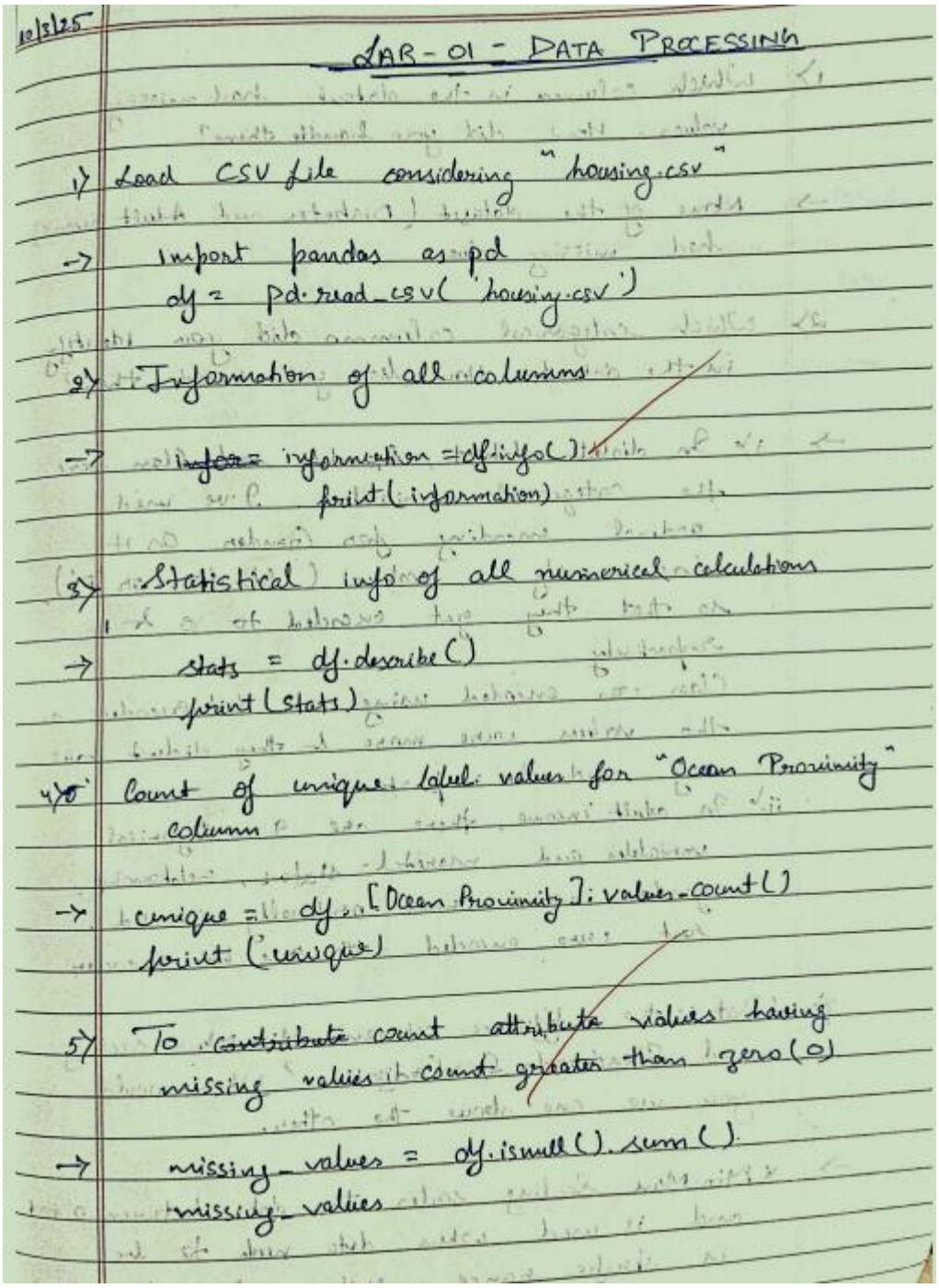
```
df_method3 = pd.read_csv('sample_sales_data.csv')
print("Method-3:")
print(df_method3.head())
print("-" * 20)
except FileNotFoundError:
print("sample_sales_data.csv not found. Please upload the file.")
print("-" * 20)
import yfinance as yf
import matplotlib.pyplot as plt
tickers = ["HDFCBANK.NS", "ICICIBANK.NS", "KOTAKBANK.NS"]
start_date = "2024-01-01"
end_date = "2024-12-30"
data = yf.download(tickers, start=start_date, end=end_date)
closing_prices = data['Close']
daily_returns = closing_prices.pct_change().dropna()
plt.figure(figsize=(12, 6))
closing_prices.plot()
plt.title('Closing Prices (2024)')
plt.xlabel('Date')
plt.ylabel('Price (INR)')
plt.grid(True)
plt.show()
plt.figure(figsize=(12, 6))
daily_returns.plot()
plt.title('Daily Returns (2024)')
plt.xlabel('Date')
```

```
plt.ylabel('Daily Return')

plt.grid(True)

plt.show()
```

Program 2

Demonstrate various data pre-processing techniques for a given dataset.

Screenshot:



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

→ None of the dataset (Diabetes and Adult income) had missing values.

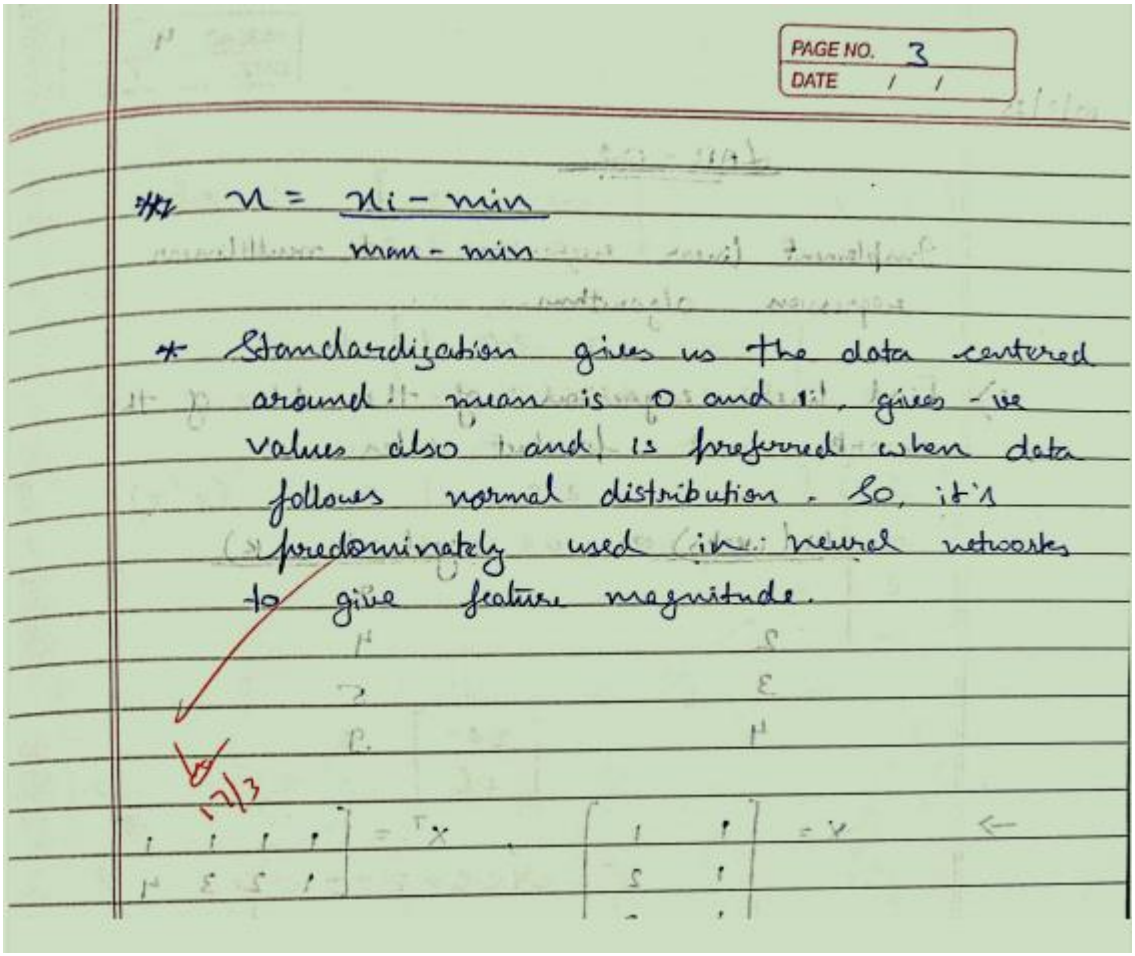
2) Which categorical columns did you identify in the datasets? How did you encode them?

→ i) In diabetes dataset, Gender and Class were the categorical variables. I've used ordinal encoding for Gender as it had only 2 values (like 'M' or 'F'), so that they get encoded to 0 & 1 respectively.
Class was encoded using one hot encoder, as the values were more & they didn't have any ordinal importance.

ii) In adult income, there are 9 categorical variables and marital-status, relationship, gender, income were ordinally encoded, rest were encoded with one hot encoder.

3) What is the difference between Min-Max Scaling and ~~Standard~~ Standardization? When would you use one above the other.

→ * Min-Max Scaling scales the data between 0 to 1 and is used when data needs to be in specific range, such as Data Learning of



Code:

```
from google.colab import files

uploaded = files.upload()


uploaded = files.upload()

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

import pandas as pd
```

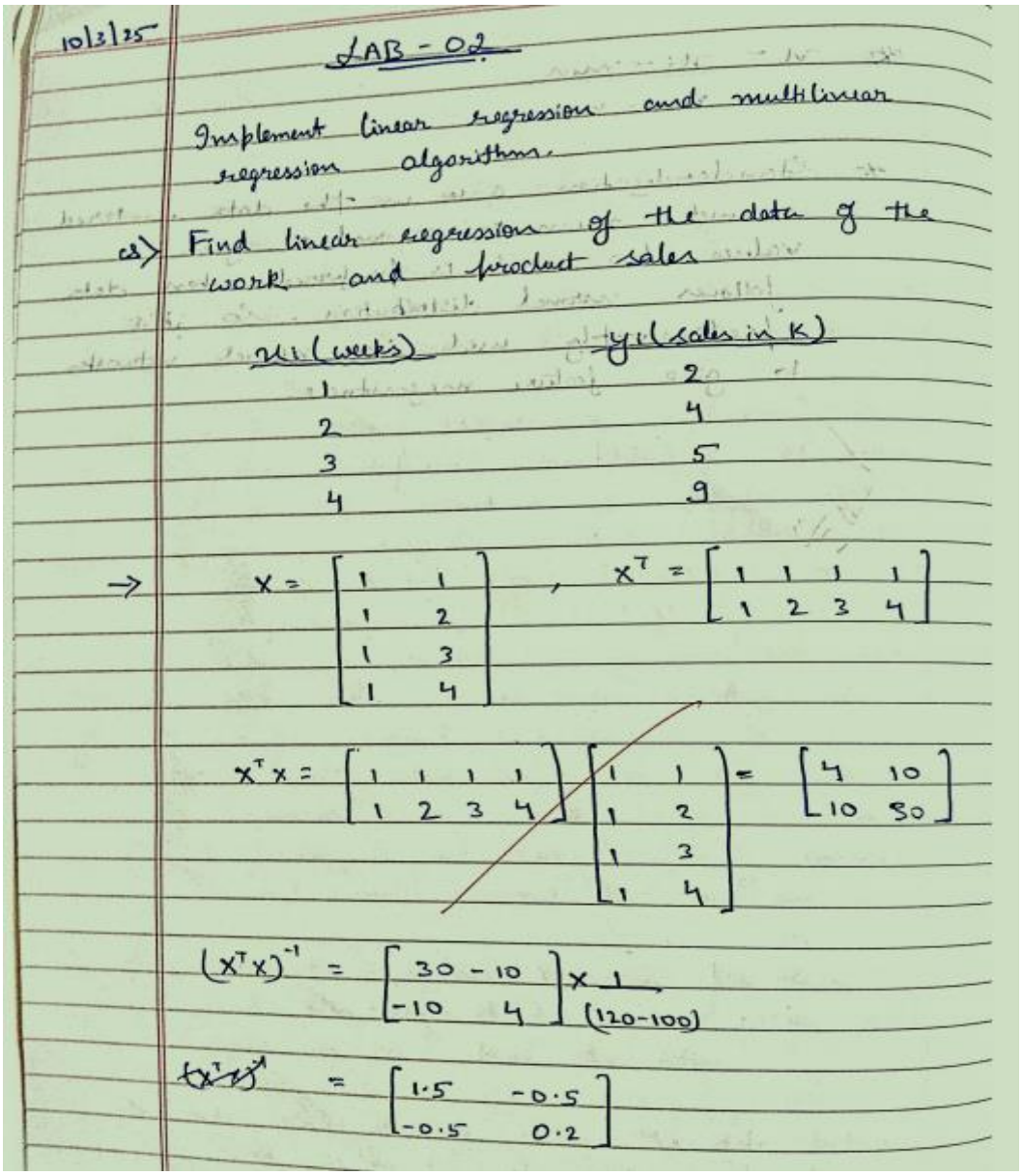
```
df1=pd.read_csv("adult.csv")
print(df1.head())
df2=pd.read_csv("Dataset of Diabetes .csv")
print(df2.head())
df1["education"].value_counts()
ordinal_encoder = OrdinalEncoder(categories=[["HS-grad",
"Some-college","Bachelors","Masters","Assoc-voc","11th","Assoc-acdm","10th","7t
h-8th","Prof-school","9th","12th","Doctorate","5th-6th","1st-4th","Preschool"]]
)
df1["Education_Encoded"] = ordinal_encoder.fit_transform(df1[["education"]])
onehot_encoder = OneHotEncoder()
encoded_data =
onehot_encoder.fit_transform(df1[["gender","relationship","workclass","occupati
on","race","native-country","income","marital-status"]])

encoded_array = encoded_data.toarray()

encoded_df = pd.DataFrame(encoded_array,
columns=onehot_encoder.get_feature_names_out(["gender","relationship","workclas
s","occupation","race","native-country","income","marital-status"]))
df_encoded = pd.concat([df1, encoded_df], axis=1)
df_encoded.drop(["education","gender","workclass","relationship","occupation","
race","native-country","income","marital-status"], 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()
df2.isnull().sum()
df2['Gender'] = df2['Gender'].replace('f', 'F')
ordinal_encoder = OrdinalEncoder(categories=[["F", "M"]])
df2["Gender_Encoded"] = ordinal_encoder.fit_transform(df2[["Gender"]])
onehot_encoder = OneHotEncoder()
encoded_data = onehot_encoder.fit_transform(df2[["CLASS"]])
encoded_array = encoded_data.toarray()
encoded_df = pd.DataFrame(encoded_array,
columns=onehot_encoder.get_feature_names_out(["CLASS"]))
df_encoded = pd.concat([df2, encoded_df], axis=1)
df2 = pd.concat([df2, encoded_df], axis=1)
df2.drop("CLASS", axis=1, inplace=True)
df2.drop("Gender", axis=1, inplace=True)
print(df2.head())
normalizer = MinMaxScaler()
df_encoded[["No_Pation","AGE","Urea","Cr" , "HbA1c" ,
"Chol","TG","HDL","LDL","VLDL","BMI"]] =
normalizer.fit_transform(df_encoded[["No_Pation","AGE","Urea","Cr" , "HbA1c" ,
"Chol","TG","HDL","LDL","VLDL","BMI"]])
df_encoded.head()
```

Program 3

Implement Linear and Multi-Linear Regression algorithm using appropriate dataset

Screenshot:



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

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

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

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

$$y = -0.5 + 2.2K$$

1) Considering the 3 data.csv files (canada_per_capita_income.csv, salary.csv, hiring.csv). Did you perform any data preprocessing steps

→ We are replacing "experience" NaN values in "experience" and "test_score(out of 10)" in "hiring.csv" using "fillna()" function. The other 2 files don't contain any missing value

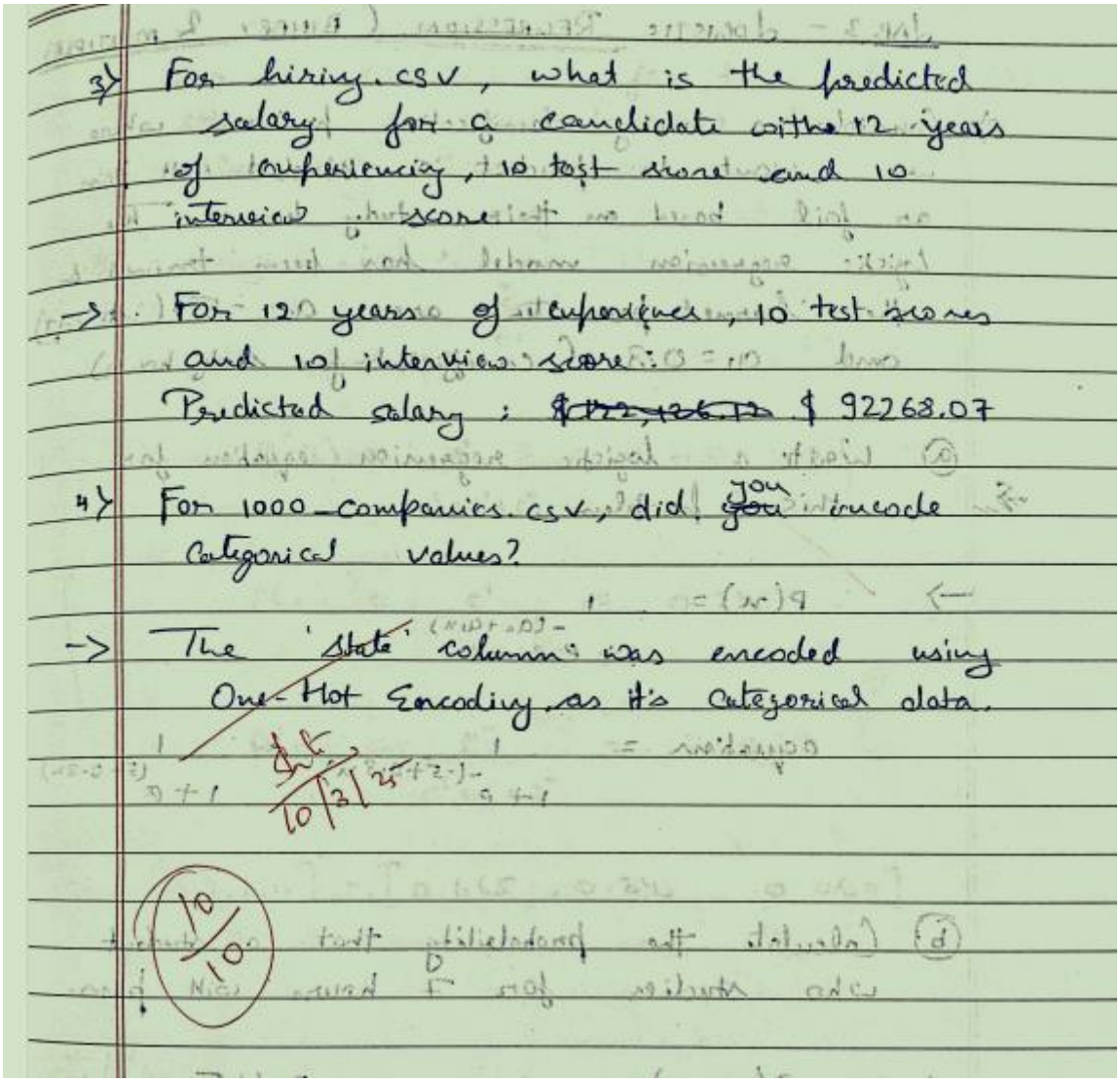
```
hiring_df["experience"] = hiring_df["experience"].fillna(0)

hiring_df["test_score(out of 10)"] = hiring_df["test_score(out of 10)"].fillna(mean)
```

2) Also encoding categorical experience values in numeric format in hiring.csv.

We also encoded "State" column in 1000 companies csv using ~~one-hot~~ OneHot encoding.

3) For canada_per_capita_income.csv. visualize the regression line with the data points? What does the plot tell you about the relationship btw year and per capital income

→ A regression line was plotted.
There is a positive linear relationship between year and per capita income, meaning increases over



Code:

```
import pandas as pd

import numpy as np

from sklearn import linear_model

import matplotlib.pyplot as plt

df = pd.read_csv('/content/housing_area_price.csv')

df

plt.xlabel('area')

plt.ylabel('price')
```

```
plt.scatter(df.area,df.price,color='red',marker='+')

new_df = df.drop('price',axis='columns')

new_df

price = df.price

price

reg = linear_model.LinearRegression()

reg.fit(new_df,price)


print(reg.coef_)

print(reg.intercept_)

reg.predict([[5000]])

df_mlr=pd.read_csv('/content/homeprices_Multiple_LR.csv')

df_mlr

df_mlr.bedrooms.median()

df_mlr.bedrooms = df_mlr.bedrooms.fillna(df_mlr.bedrooms.median())

print(df_mlr)

reg = linear_model.LinearRegression()

reg.fit(df_mlr.drop('price',axis='columns'),df_mlr.price)

reg.coef_

reg.intercept_

#Find price of home with 3000 sqr ft area, 3 bedrooms, 40 year old

reg.predict([[3000, 3, 40]])

112.06244194*3000 + 23388.88007794*3 + -3231.71790863*40 + 221323.00186540384


canada=pd.read_csv('/content/canada_per_capita_income.csv')

canada
```

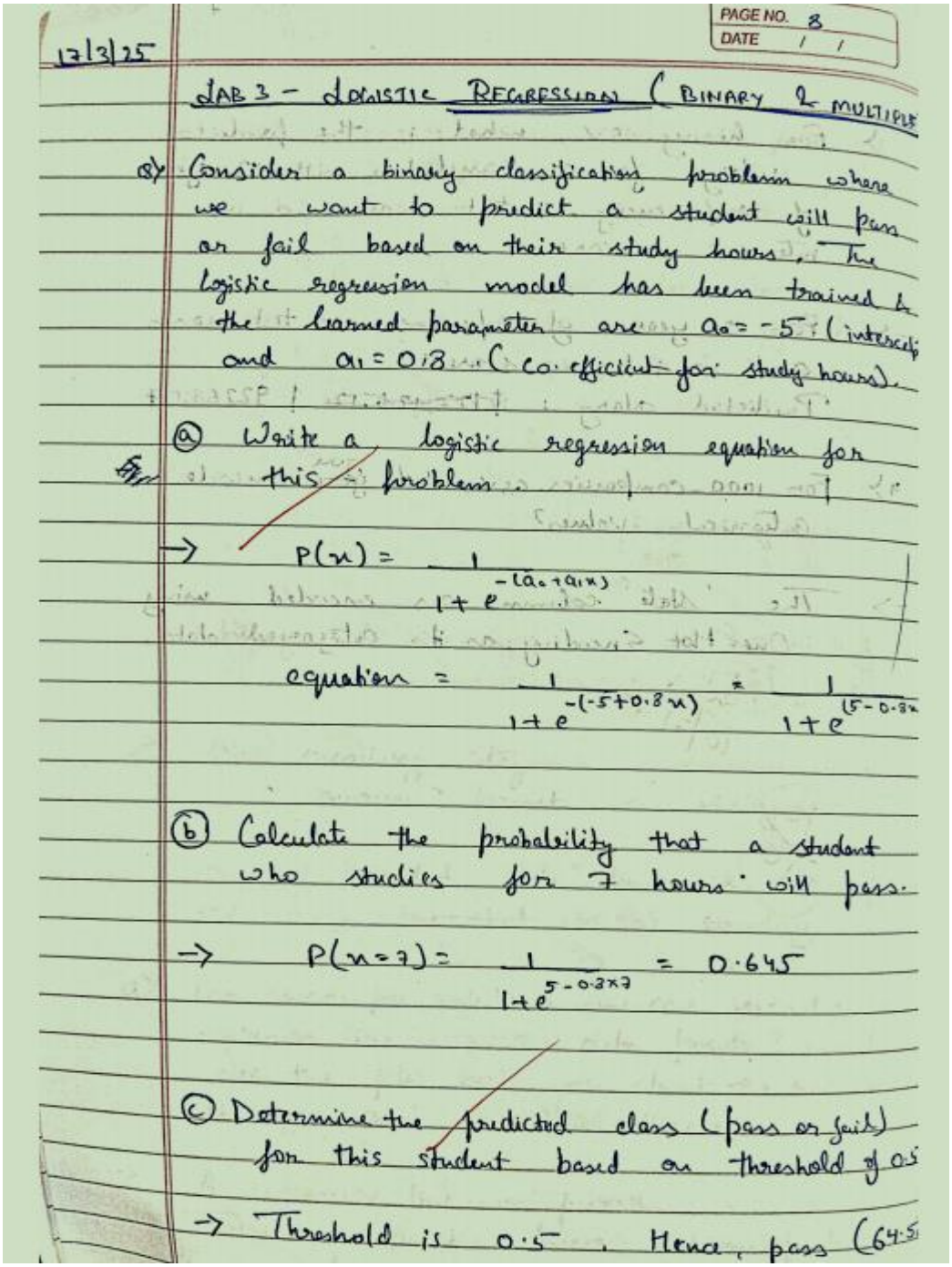
```
plt.xlabel('year')
plt.ylabel('per capita income (US$)')
canada.rename(columns={'per capita income (US$)':'income'},inplace=True)
plt.scatter(canada.year,canada.income ,color='red',marker='+')
new_df_canada = canada.drop('income',axis='columns')
new_df_canada.sample(5)
income = canada.income
income.sample(5)
reg = linear_model.LinearRegression()
reg.fit(new_df_canada,income)
print(reg.coef_)
print(reg.intercept_)
reg.predict([[2020]])
hiring=pd.read_csv('/content/hiring.csv')
hiring
hiring['experience'].fillna(0, inplace=True)
hiring['test_score(out of 10)'].fillna(hiring['test_score(out of 10)'].mean(),
inplace=True)

def convert_to_int(word):
word_dict = {'one':1, 'two':2, 'three':3, 'four':4, 'five':5, 'six':6,
'seven':7, 'eight':8,
'nine':9, 'ten':10, 'eleven':11, 'twelve':12, 'zero':0, 0: 0}
return word_dict[word]
hiring['experience'] = hiring['experience'].apply(lambda x : convert_to_int(x))
```

```
hiring

reg = linear_model.LinearRegression()

hiring.rename(columns={'salary($)':'salary'},inplace=True)

reg.fit(hiring.drop('salary',axis='columns'),hiring.salary)

print(reg.coef_)

print(reg.intercept_)

#What is the predicted salary for a candidate with 12 years of experience, 10

test score, and 10 interview score?

reg.predict([[12, 10, 10]])


comp=pd.read_csv('/content/1000_Companies.csv')

comp

comp.isnull().sum()

from sklearn.preprocessing import OneHotEncoder

ohe = OneHotEncoder(sparse_output=False, handle_unknown='ignore')

state_encoded = ohe.fit_transform(comp[['State']])

state_encoded_df = pd.DataFrame(state_encoded,

columns=ohe.get_feature_names_out(['State']))

comp = pd.concat([comp, state_encoded_df], axis=1).drop(columns=['State'])

print(comp)

reg = linear_model.LinearRegression()

reg.fit(comp.drop('Profit',axis='columns'),comp.Profit)

print(reg.coef_)

print(reg.intercept_)


reg.predict([[91694.48, 515841.3, 11931.24,0,1,0]])
```

Program 4

Build Logistic Regression Model for a given dataset.

Screenshot:



17/3/25

PAGE NO. 8
DATE / /

LAB 3 - LOGISTIC REGRESSION (BINARY & MULTIPLE

Q) Consider a binary classification problem where we want to predict a student will pass or fail based on their study hours. The logistic regression model has been trained & the learned parameter are $a_0 = -5$ (intercept) and $a_1 = 0.8$ (Co-efficient for study hours).

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

→ $P(x) = \frac{1}{1 + e^{-(a_0 + a_1 x)}}$

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

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

→ $P(x=7) = \frac{1}{1 + e^{5 - 0.8 \times 7}} = 0.645$

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

→ Threshold is 0.5. Hence, pass (64.5

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

→ $$\text{Softmax}(z) = \frac{e^z}{\sum_{i=1}^{n} e^i}$$

$$P(2) = \frac{e^2}{e^2+e^1+e^0} = 0.665$$

$$P(1) = \frac{e^1}{e^2+e^1+e^0} = 0.245$$

$$P(0) = \frac{e^0}{e^2+e^1+e^0} = 0.090$$

$$P[2,1,0] = [0.665, 0.245, 0.090]$$

3) For dataset file, "HR_comma_sep.csv".

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

→ The key variables are:

- Satisfaction level → Employees with lower satisfaction levels are likely to leave the comp
- no_of_project → overiwork is more likely to leave
- Time spent in the company → Employees who spent more time in the comp are more likely to leave.
- Salary → Employees with lower salary have higher attrition rate.

ii) What was the accuracy of your logistic regression model? Do you think this is a good accuracy? Why or Why not?

→ Accuracy of the model : 74.83%.

The accuracy is fairly good for classification problems but not ideal.

accuracy = accuracy_score (y_test, y_pred)

4) For Zoo dataset,

i) Did you perform any data preprocessing steps? If yes, then what were they and why were they necessary?

→ • Dropped animal_name column as it's not a useful feature for classification.

• ~~No missing value was found.~~

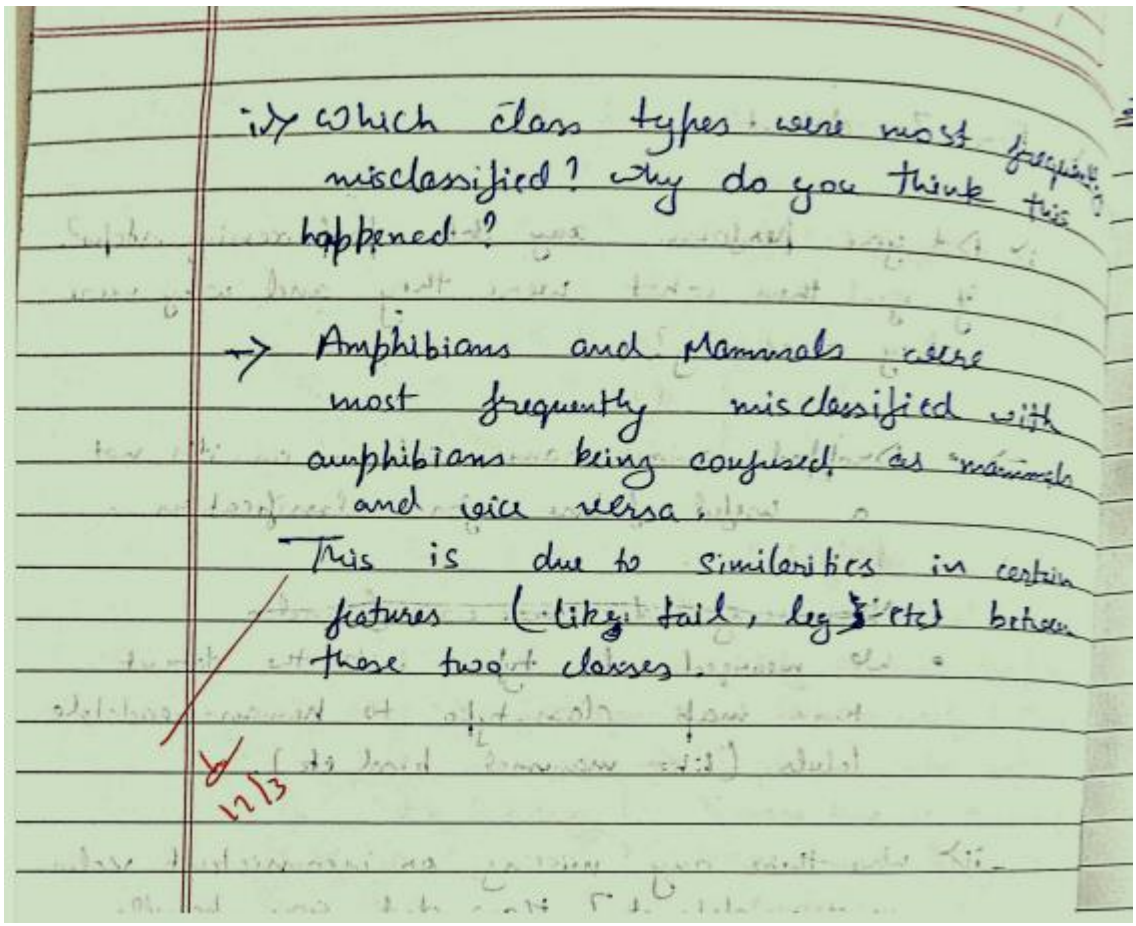
• We mearged class types with the dataset to map class_type to human-readable labels (like mammal, bird etc).

ii) Were there any missing or inconsistent values in the dataset? How did you handle them?

→ No missing values were found in the dataset. The data was clean and ready for modelling.

iii) What does the confusion matrix tell you about the performance of your ~~one~~ model?

→ The confusion matrix shows that the model performs well, with most predictions being correct.
Some misclassification like amphibians and mammals, suggest there may be overlap in the features between those classes.



Code:

```
import pandas as pd

from matplotlib import pyplot as plt

from seaborn import regplot

import seaborn as sns

df1=pd.read_csv('HR_comma_sep.csv')

df1.sample(10)

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()
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()
from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder
import numpy as np
import seaborn as sns
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()
correlation_threshold = 0.1
correlated_features = correlation_matrix['left'].abs() > correlation_threshold
highly_correlated_features =
correlated_features[correlated_features].index.tolist()
new_df = df1[highly_correlated_features]
print(new_df.head())

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

X = new_df.drop('left', axis=1)
y = new_df['left']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
example_data = X_test.iloc[0].values.reshape(1, -1)
prediction = model.predict(example_data)

import pandas as pd
```

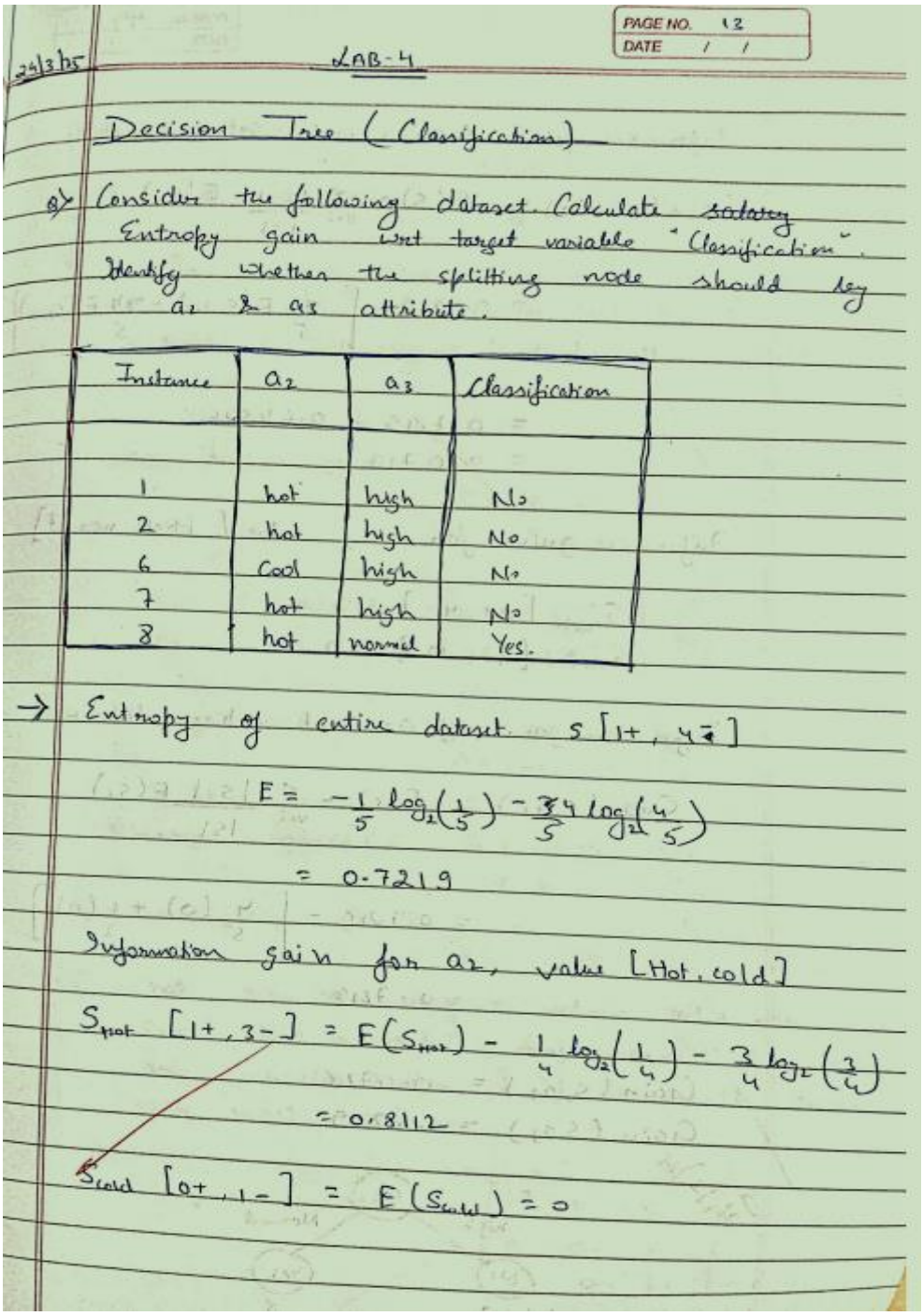
```
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score

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}")


from sklearn.metrics import confusion_matrix

import seaborn as sns
```

```
import matplotlib.pyplot as plt

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 5

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

Screenshot:



24/3/25 LAB-4

PAGE NO. 12
DATE / /

Decision Tree (Classification)

Q) Consider the following dataset. Calculate ~~Entropy~~ Entropy gain wrt target variable "Classification". Identify whether the splitting node should be a_2 & 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.

→ Entropy of entire dataset $S[1+, 4-]$

$$E = -\frac{1}{5}\log_2\left(\frac{1}{5}\right) - \frac{4}{5}\log_2\left(\frac{4}{5}\right)$$

$$= 0.7219$$

Information gain for a_2, value [Hot, cold]

$$S_{hot}[1+, 3-] = E(S_{hot}) - \frac{1}{4}\log_2\left(\frac{1}{4}\right) - \frac{3}{4}\log_2\left(\frac{3}{4}\right)$$

$$= 0.8112$$

$$S_{cold}[0+, 1-] = E(S_{cold}) = 0$$

$Gain(S, a_2) = E(S) - \sum_{v=i} \frac{|S_v|}{|S|}$

$= 0.7219 - \left[\frac{5}{5} E(S_{weak}) + \frac{4}{5} E(S_{strong}) \right]$

$= 0.7219 - 0.64396$

$= 0.0728$

Information gain for a_3 value [High, normal]

$S_{high}[0+, 4-] = 0$

$S_{normal}[4+, 0] = 0$

Information gain of a_3 wrt entire dataset

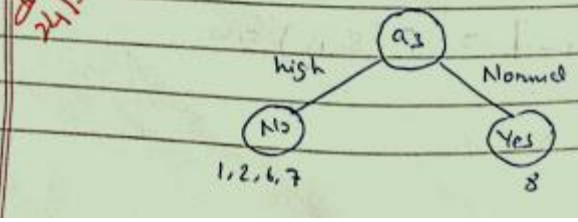
$$Gain(S, a_3) = E(S) - \sum_{v=i} \frac{|S_v|}{|S|} E(S_v)$$

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

$= 0.7219$

$Gain(S, a_2) = 0.0728$

$Gain(S, a_3) = 0.2219$



1) For "iris.csv" dataset

(a) What was the accuracy score for the 'iris' dataset?

→ The accuracy score for the iris dataset is 1 (100%), which indicates that the model correctly classified all the test samples.

Confusion matrix : $\begin{bmatrix} 10 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 11 \end{bmatrix}$

Accuracy (setosa) = $\frac{10 + (11+9)}{10+11+9} = 1$

Accuracy (versicolor) = $\frac{9 + (10+11)}{10+9+11} = 1$

Accuracy (virginica) = $\frac{11 + (10+9)}{10+9+11} = 1$

(b) What does the confusion matrix tell us about the model's performance? Were there any misclassification? If so, which classes were most confused.

→ Confusion matrix : $\begin{bmatrix} 10 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 11 \end{bmatrix}$

The rows represent the actual class and columns represent predicted class. The diagonal values (10, 9, 11) represent correctly classified instance for each class. All values outside the diagonal element are misclassifications. Here, all values outside other than diagonal elements are 0, meaning there were no misclassifications.

Since, no misclassification occured, no classes were confused.

2) For "petrol consumption.csv" dataset

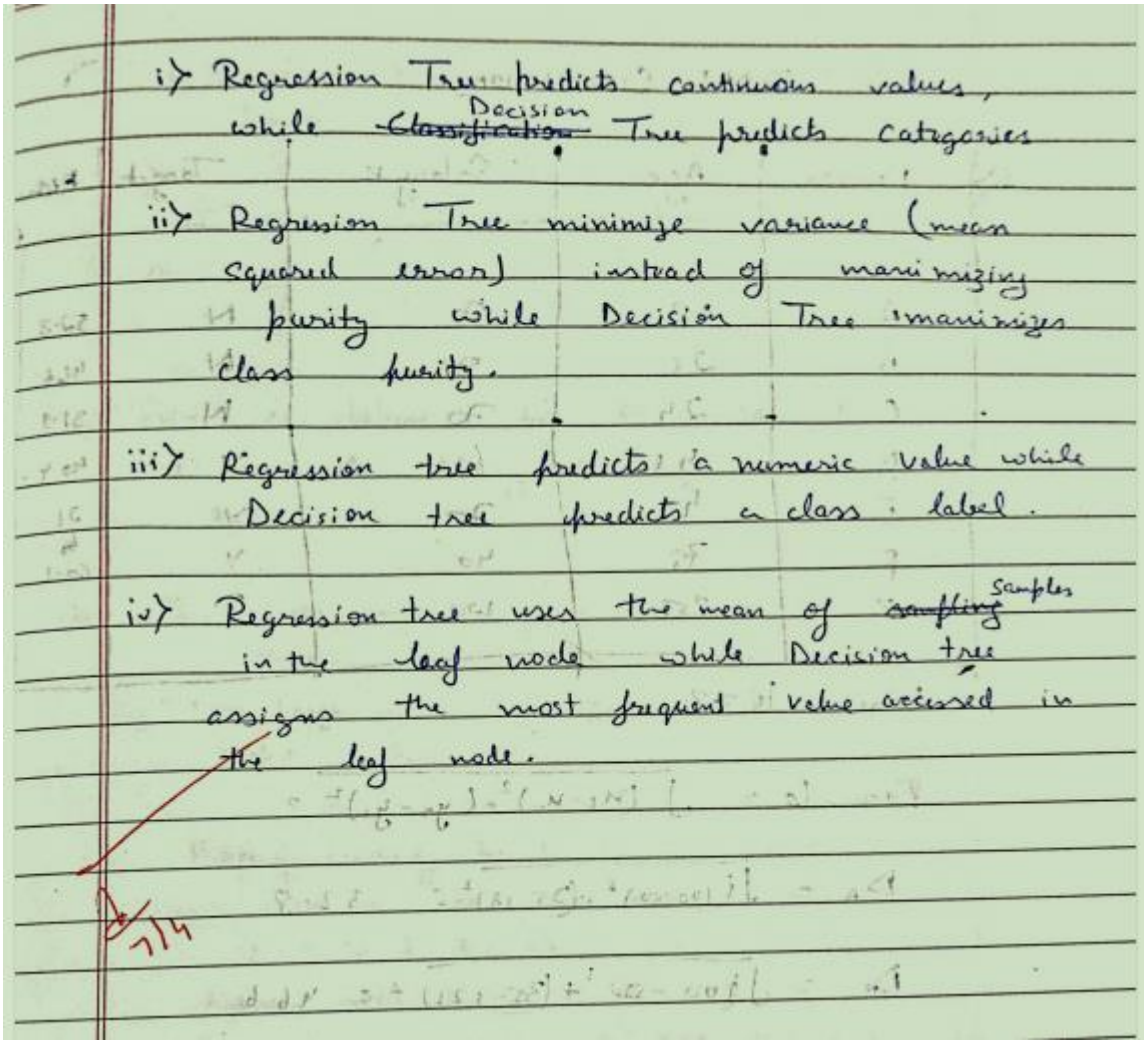
(a) Can you interpret the Regression Tree? What are the most important features for predicting petrol consumption?

→ i) Population driver's license → More drivers leads to more petrol consumption.

ii) Income per capita (Avg. income) → Higher income may lead to more vehicle usage.

iii) ~~Road length~~ Paved Highway (km) → More roads may correlate with increased fuel demand.

(b) How does the Regression Tree handle continuous target variables compared to the Decision Tree classifier.



Code:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
data = {
'a1': [True, True, False, False, False, True, True, True, False, False],
'a2': ['Hot', 'Hot', 'Hot', 'Cool', 'Cool', 'Cool', 'Hot', 'Hot', 'Cool',
'Cool'],
```

```
'a3': ['High', 'High', 'High', 'Normal', 'Normal', 'High', 'High',
'Normal', 'Normal', 'High'],
'Classification': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'No', 'Yes',
'Yes', 'Yes']
}
data
df = pd.DataFrame(data)
label_encoders = {}
for column in df.columns:
le = LabelEncoder()
df[column] = le.fit_transform(df[column])
label_encoders[column] = le
df
X = df.drop('Classification', axis=1)
y = df['Classification']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
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, target_names=['No', 'Yes']))
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(12,8))
plot_tree(clf, filled=True, feature_names=X.columns, class_names=['No', 'Yes'])
plt.show()


iris=pd.read_csv("/content/iris - Copy.csv")
iris
le = LabelEncoder()
iris["species"] = le.fit_transform(iris["species"])
label_encoders[column] = le
iris
X = iris.drop('species', axis=1)
y = iris['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, target_names=['Iris-setosa',
'Iris-versicolor','Iris-virginica',]))
from sklearn.metrics import confusion_matrix
import seaborn as sns
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
```

```
xticklabels=['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'],
yticklabels=['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
plt.figure(figsize=(12,8))
plot_tree(clf, filled=True, feature_names=X.columns,
class_names=['Iris-setosa', 'Iris-versicolor','Iris-virginica'])
plt.show()


drug=pd.read_csv("/content/drug - Copy.csv")
print(drug)
drug["Drug"].unique()
le = LabelEncoder()
drug["Sex"] = le.fit_transform(drug["Sex"])
drug["BP"] = le.fit_transform(drug["BP"])
drug["Cholesterol"] = le.fit_transform(drug["Cholesterol"])
#drug["Drug"] = le.fit_transform(drug["Drug"])
label_encoders[column] = le
drug
X = drug.drop('Drug', axis=1)
y = drug['Drug']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

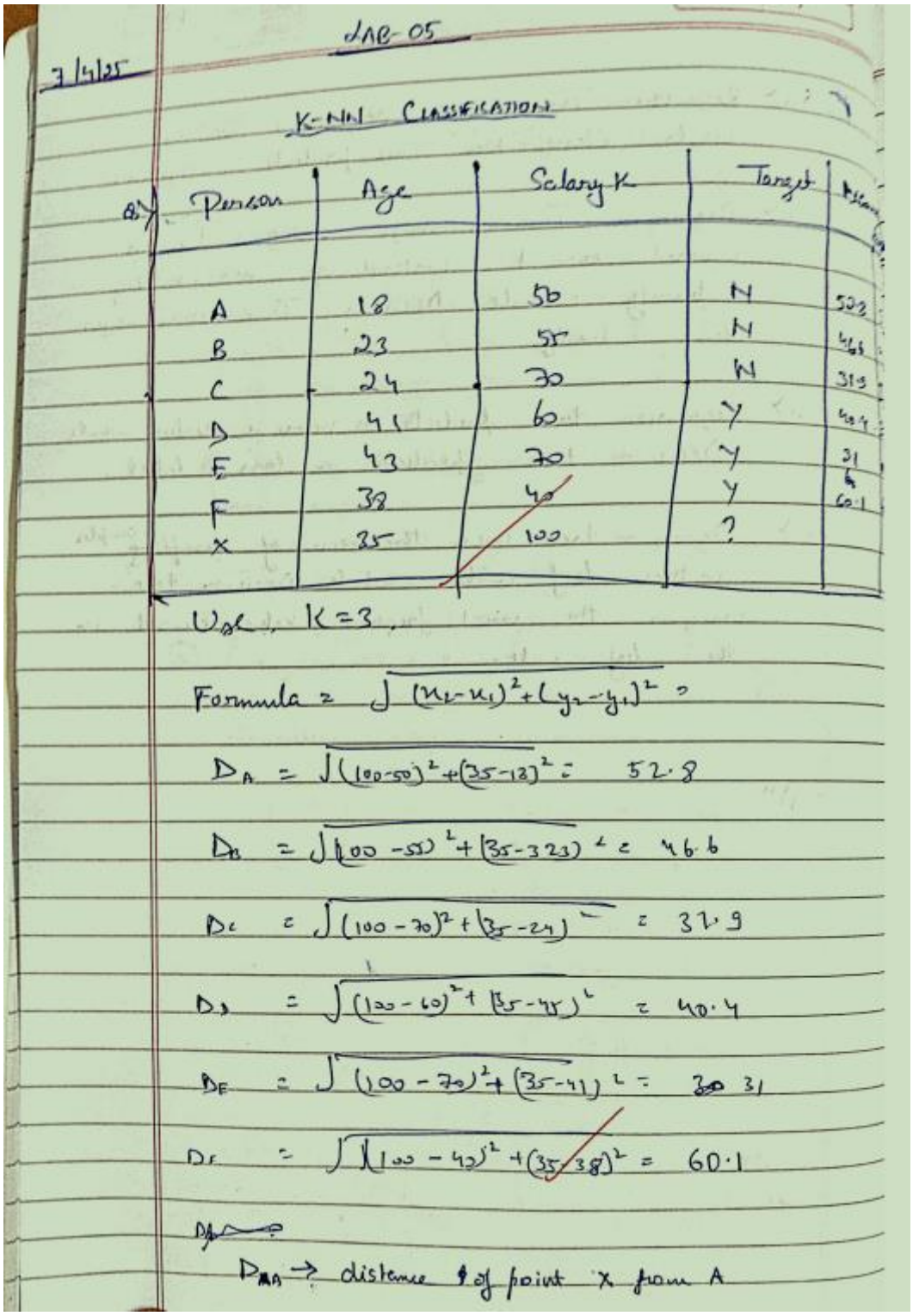
```
clf = DecisionTreeClassifier(criterion='entropy')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
print(classification_report(y_test, y_pred, target_names=['drugY', 'drugC',
'drugX', 'drugA', 'drugB']))
from sklearn.metrics import confusion_matrix
import seaborn as sns
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=['drugY', 'drugC', 'drugX', 'drugA', 'drugB'],
yticklabels=['drugY', 'drugC', 'drugX', 'drugA', 'drugB'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

petrol=pd.read_csv("/content/petrol_consumption - Copy.csv")
petrol
X = petrol.drop('Petrol_Consumption', axis=1)
y = petrol['Petrol_Consumption']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```
clf = DecisionTreeRegressor()

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

from sklearn.tree import DecisionTreeRegressor

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_absolute_error, mean_squared_error


mae = mean_absolute_error(y_test, y_pred)

mse = mean_squared_error(y_test, y_pred)

rmse = mean_squared_error(y_test, y_pred)**0.5

print(f'Mean Absolute Error: {mae}')

print(f'Mean Squared Error: {mse}')

print(f'Root Mean Squared Error: {rmse}')

plt.figure(figsize=(30, 30))

plot_tree(clf, filled=True, feature_names=X.columns, fontsize=10)

plt.show()
```

Program 6

Build KNN Classification model for a given dataset.

Screenshot:



Nearest 3 neighbours are C, D, E and their target is N, Y, Y respectively

Hence, X will be Y. (target variable for X)

a) For iris dataset, how to choose the k value? Demonstrate using accuracy rate and error rate

→ Too small ⇒ very sensitive to noise value

Too large ⇒ may ignore local pattern value

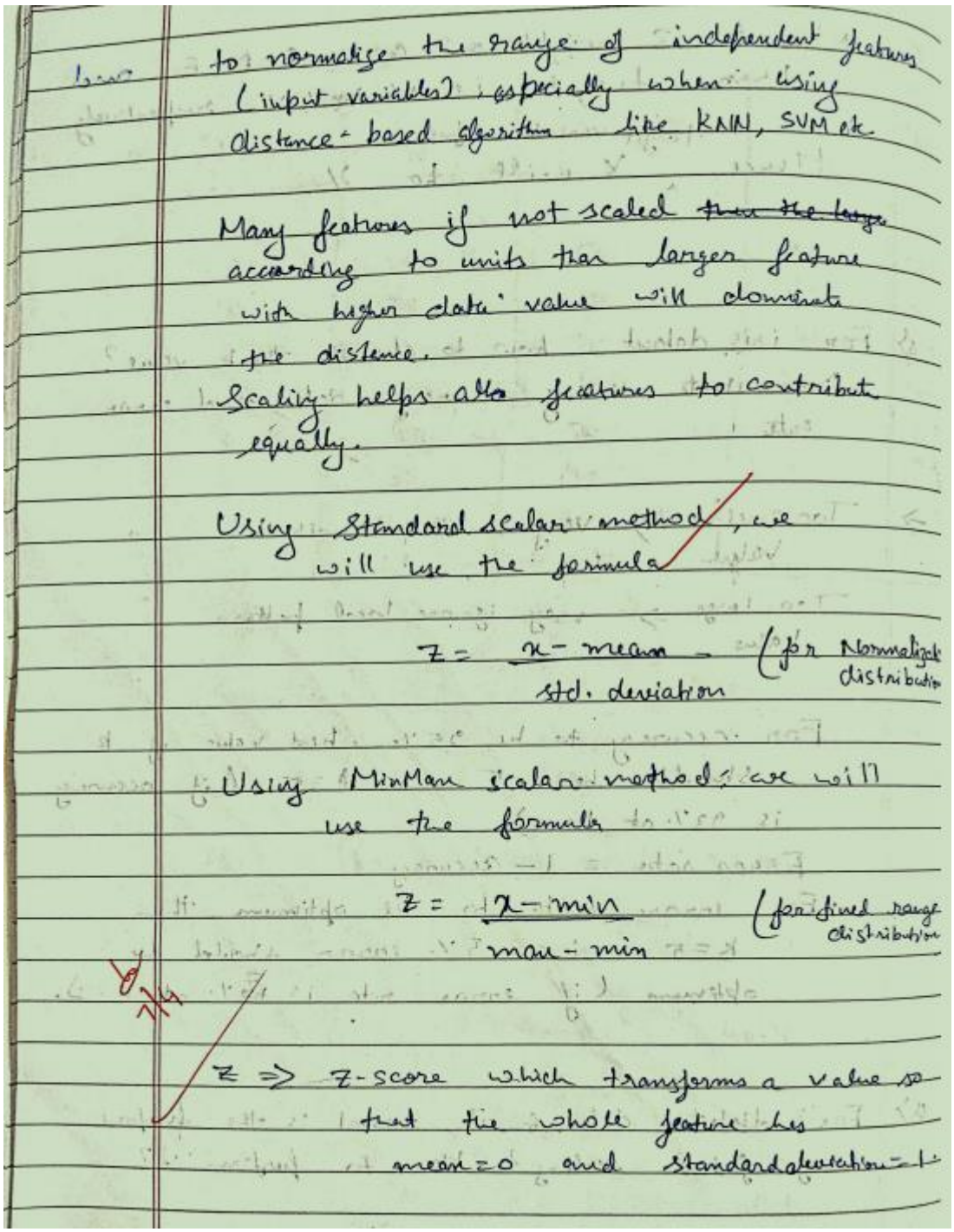
For accuracy to be 95%, best value of k should be 5 i.e. k=5 (if accuracy is 93% at k=3).

Error rate = 1 − Accuracy

For error rate to be optimum, then k=5 i.e. 5% error should be optimum (if error rate is 7% at k=3).

b) For diabetes dataset, what is the purpose of feature scaling? How to perform it?

→ Feature scaling is good for diabetes dataset,



Code:

```
import pandas as pd

iris=pd.read_csv('iris.csv')

iris.head()

from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier
```

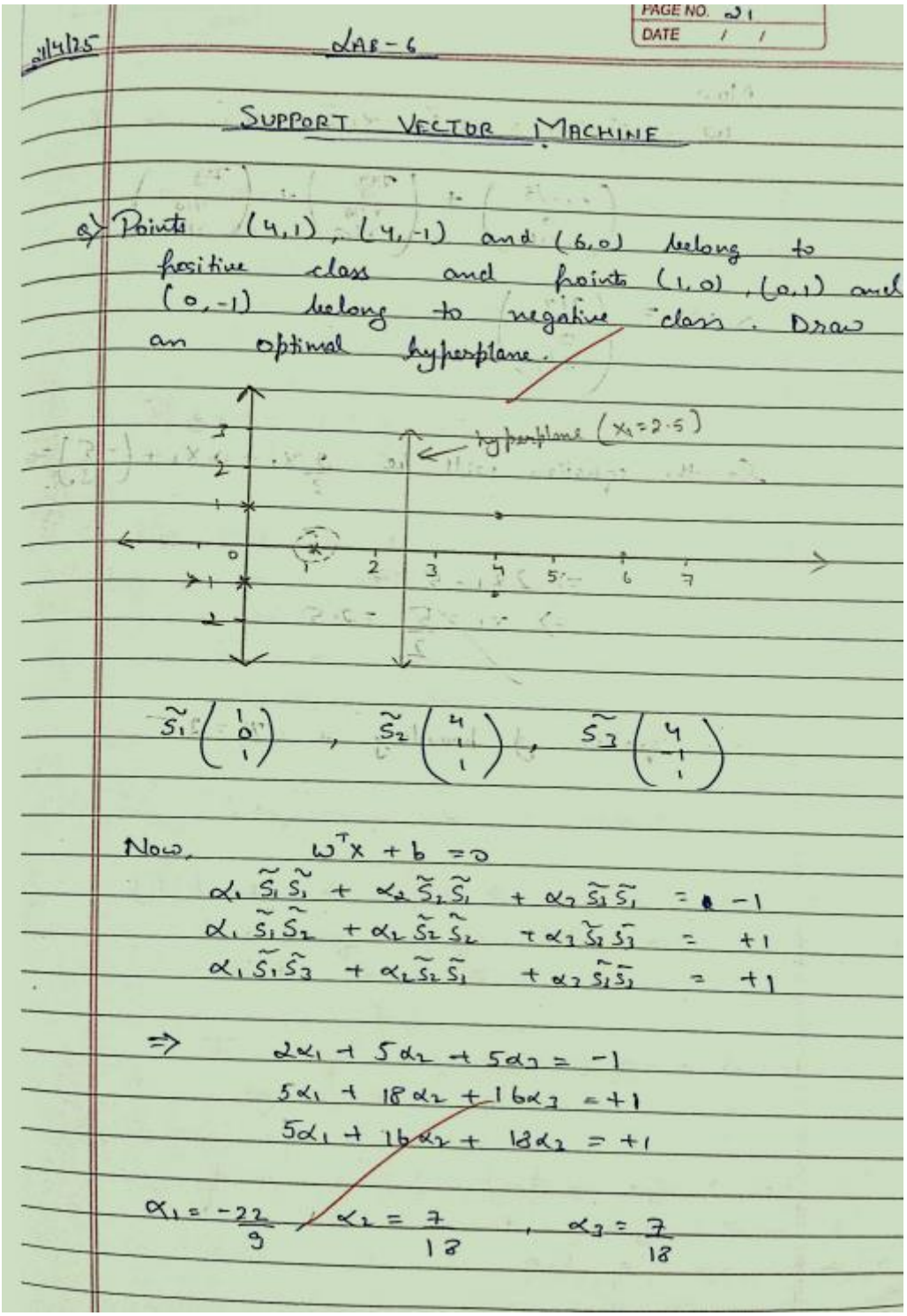
```
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns

X = iris.drop('species', axis=1)
y = iris['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=best_k)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

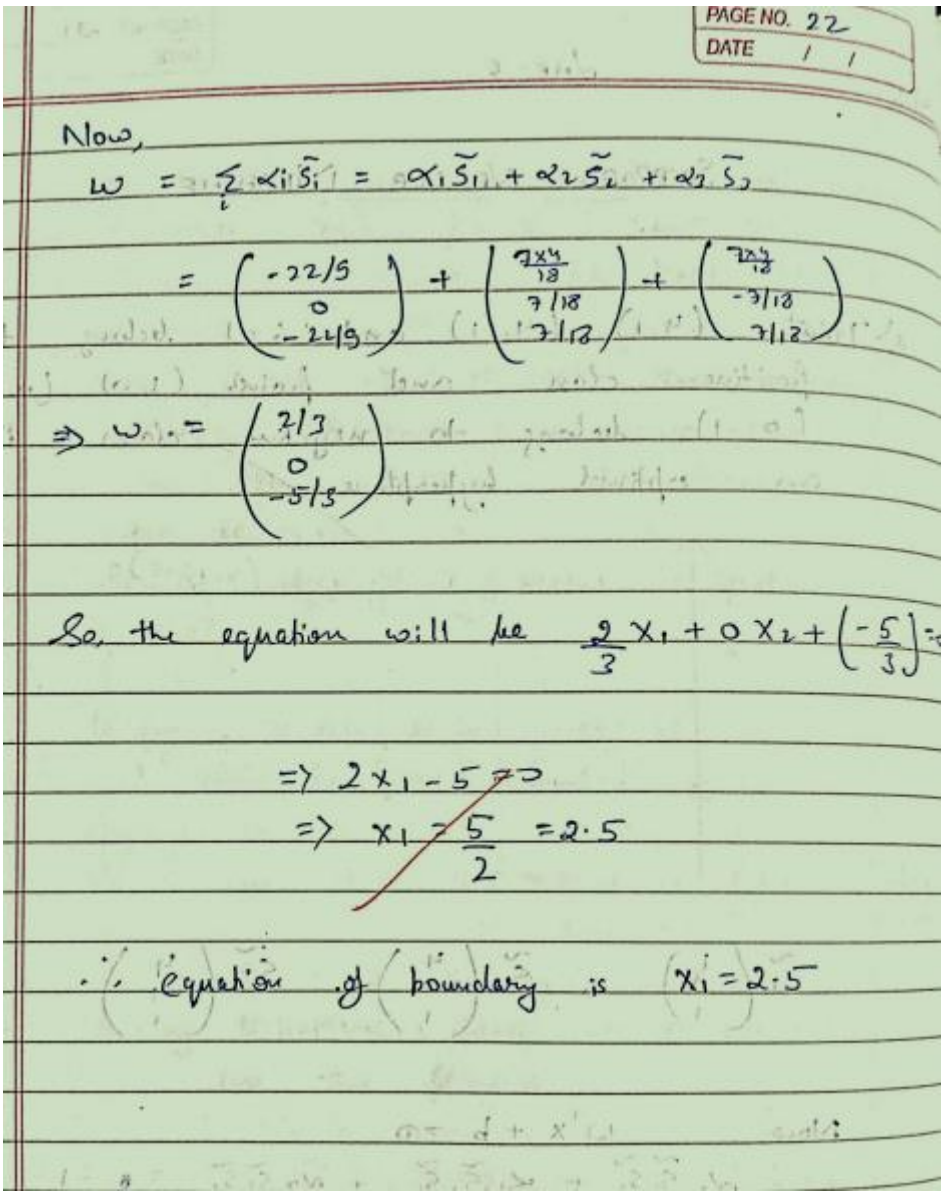
```
accuracy = accuracy_score(y_test, y_pred)

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',xticklabels=iris['species'].unique(),

yticklabels=iris['species'].unique())

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.title("Confusion Matrix")

plt.show()


print("\nClassification Report:")

print(classification_report(y_test, y_pred))
```

Program 7

Build Support vector machine model for a given dataset

Screenshot:




PAGE NO. 22
DATE / /

Now,

$$w = \sum_i \alpha_i \tilde{s}_i = \alpha_1 \tilde{s}_1 + \alpha_2 \tilde{s}_2 + \alpha_3 \tilde{s}_3$$

$$= \begin{pmatrix} -22/5 \\ 0 \\ -22/5 \end{pmatrix} + \begin{pmatrix} \frac{7\times4}{18} \\ 7/18 \\ 7/18 \end{pmatrix} + \begin{pmatrix} \frac{7\times4}{18} \\ -7/18 \\ 7/18 \end{pmatrix}$$

$$\Rightarrow w_0 = \begin{pmatrix} 7/3 \\ 0 \\ -5/3 \end{pmatrix}$$

So, the equation will be $\frac{2}{3}x_1 + 0\,x_2 + \left(-\frac{5}{3}\right) = 0$

$$\Rightarrow 2x_1 - 5 = 0$$

$$\Rightarrow x_1 = \frac{5}{2} = 2.5$$

$\therefore$ equation of boundary is $x_1 = 2.5$

1) For "iris.csv" dataset,

(a) What is the accuracy score of the classifier using the linear kernel and RBF kernel?

→ Linear kernel : 1.0 (100%)
RBF kernel : 1.0 (100%)

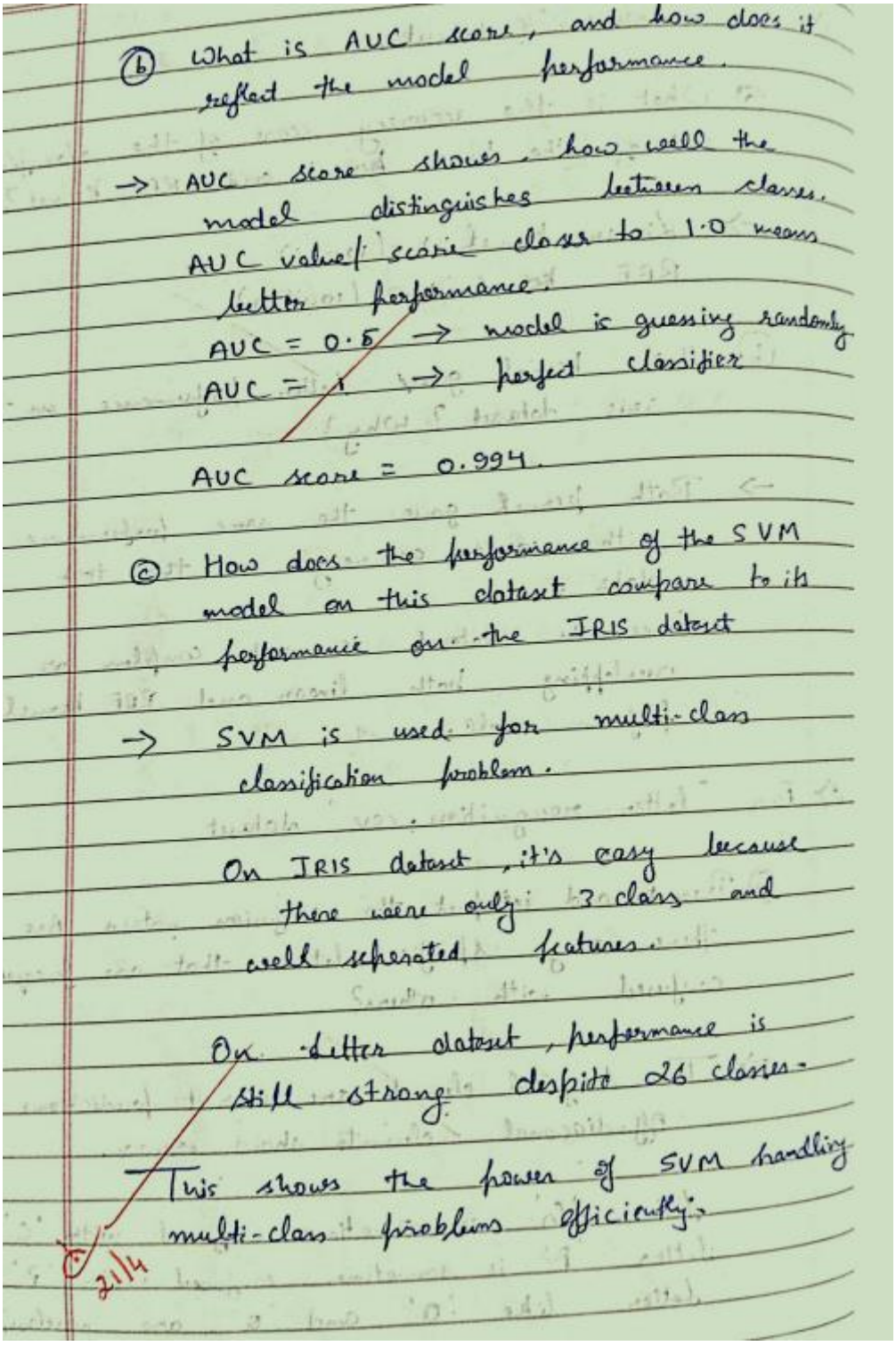
(b) Which kernel gave better performance on the iris dataset? Why?

→ Both kernel gave the same performance with 100% accuracy on the test data.
Since the dataset is not complex or overlapping, both linear and RBF kernel perform well.

2) For "letter-recognition.csv" dataset,

(a) Present and interpret the confusion matrix. Are there any specific letters that are frequently confused with others?

→ The diagonal elements are correct predictions.
Off-diagonal elements show errors.
Letter 'G' is sometimes confused with 'C' & 'Q'
Letter 'F' is sometimes confused with 'P'
Letter like 'O' and 'Q' are misclassified.



Code:

```
import pandas as pd

iris=pd.read_csv('iris.csv')

iris.head()

iris.isnull().sum()
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
iris = pd.read_csv('iris.csv')
X = iris.drop('species', axis=1)
y = iris['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
svm_rbf = SVC(kernel='rbf', gamma='scale')
svm_rbf.fit(X_train, y_train)
y_pred_rbf = svm_rbf.predict(X_test)
accuracy_rbf = accuracy_score(y_test, y_pred_rbf)
print(f"Accuracy (RBF Kernel): {accuracy_rbf}")
cm_rbf = confusion_matrix(y_test, y_pred_rbf)
plt.figure(figsize=(8, 6))
sns.heatmap(cm_rbf, annot=True, fmt="d", cmap="Blues",
xticklabels=iris['species'].unique(),
yticklabels=iris['species'].unique())
plt.title("Confusion Matrix (RBF Kernel)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
svm_linear = SVC(kernel='linear')
```

```
svm_linear.fit(X_train, y_train)
y_pred_linear = svm_linear.predict(X_test)
accuracy_linear = accuracy_score(y_test, y_pred_linear)
print(f"Accuracy (Linear Kernel): {accuracy_linear}")
cm_linear = confusion_matrix(y_test, y_pred_linear)
plt.figure(figsize=(8, 6))
sns.heatmap(cm_linear, annot=True, fmt="d", cmap="Blues",
xticklabels=iris['species'].unique(),
yticklabels=iris['species'].unique())
plt.title("Confusion Matrix (Linear Kernel)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()


df2=pd.read_csv('letter-recognition.csv')
df2.head()
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelBinarizer
X = df2.drop('letter', axis=1)
y = df2['letter']
```

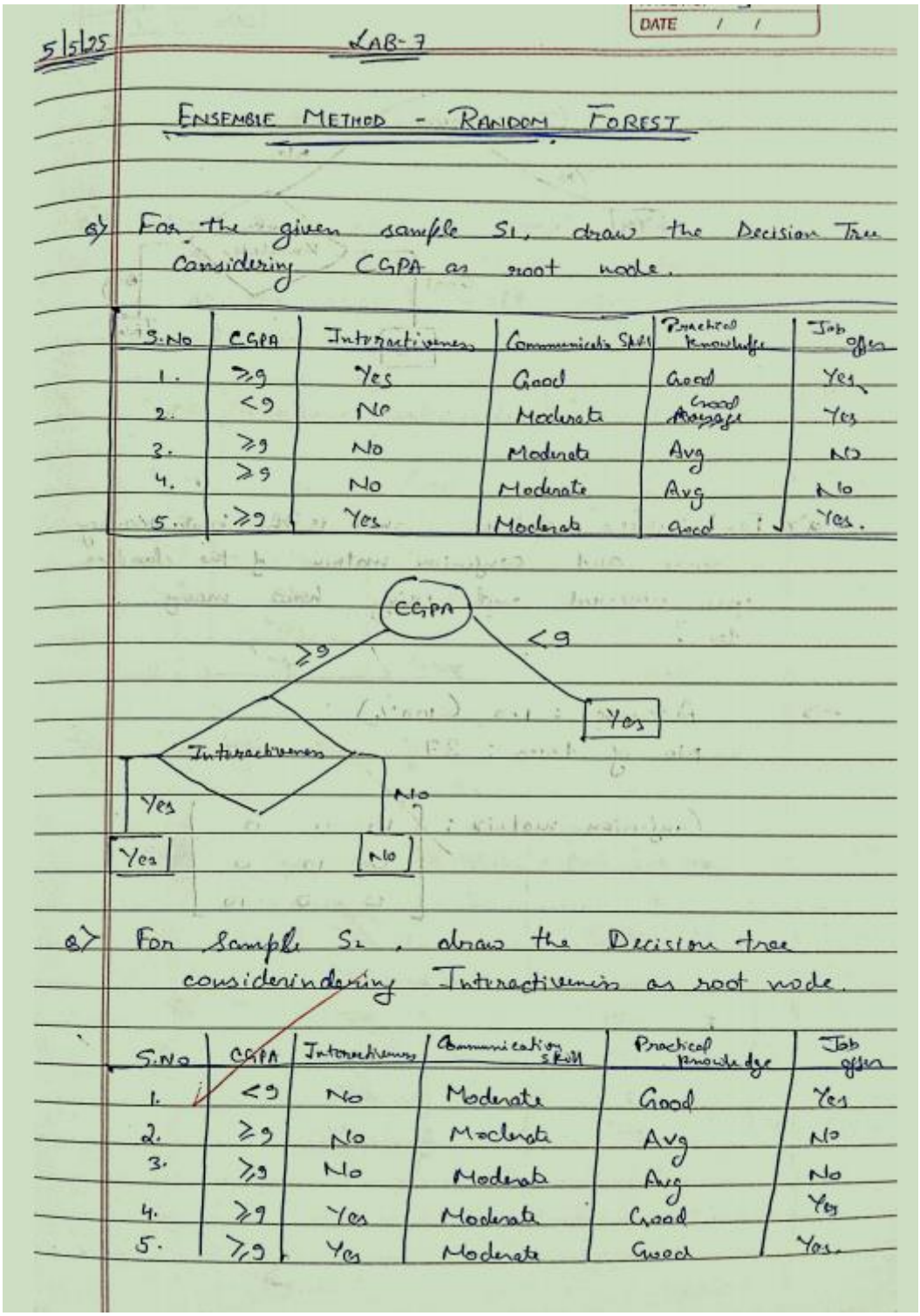
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
svm_classifier = SVC(kernel='linear', probability=True)
svm_classifier.fit(X_train, y_train)

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)
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:



5/5/25

LAB-7

DATE / /

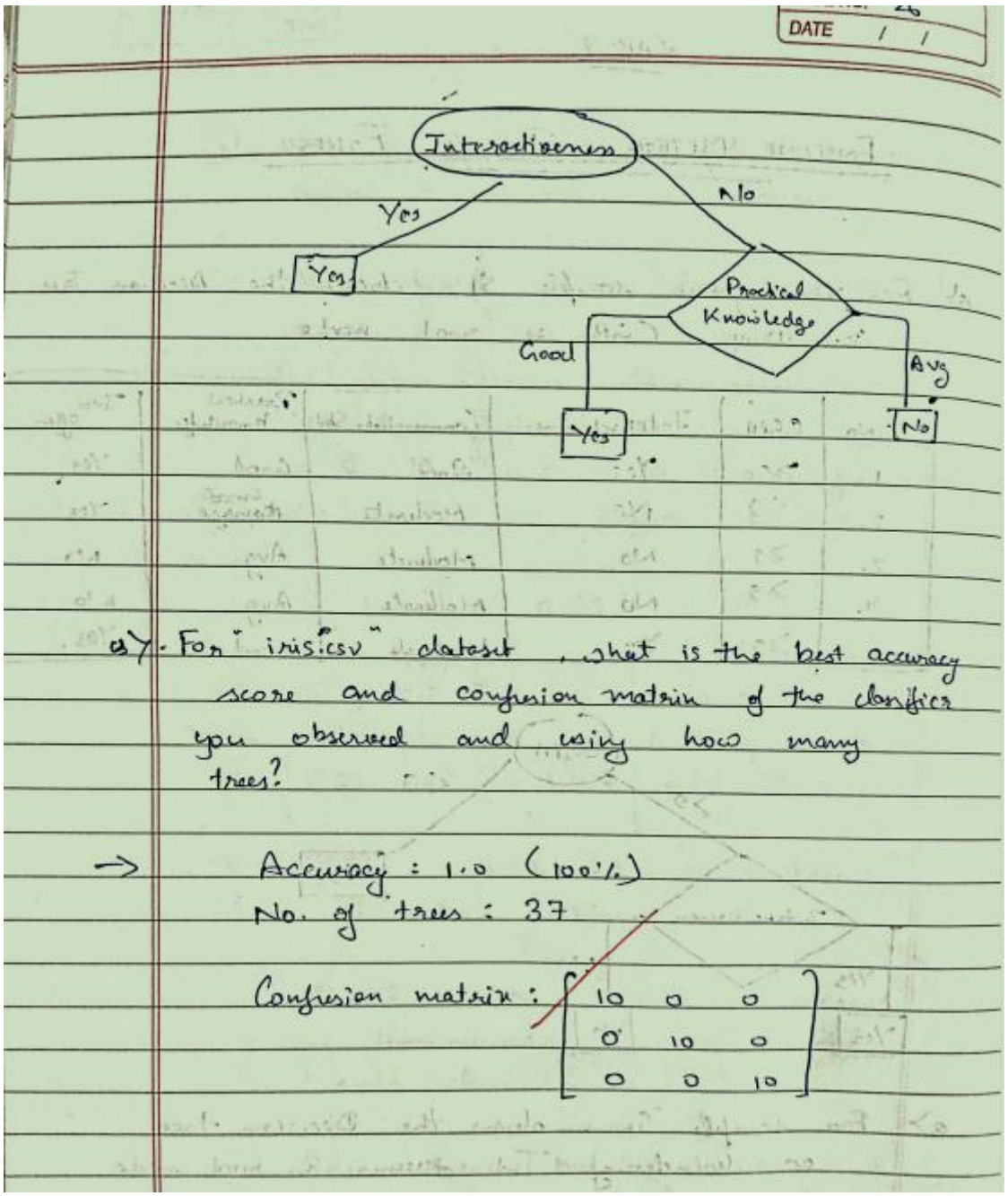
ENSEMBLE METHOD - RANDOM FOREST

a) For the given sample S1, draw the Decision Tree considering CGPA as root node.

S.No	CGPA	Interactiveness	Communicative Skill	Practical Knowledge	Job offer
1.	≥9	Yes	Good	Good	Yes
2.	<9	No	Moderate	Good Average	Yes
3.	≥9	No	Moderate	Avg	No
4.	≥9	No	Moderate	Avg	No
5.	≥9	Yes	Moderate	Good	Yes.

a) For Sample S2, draw the Decision tree considering Interactiveness as root node.

S.No	CGPA	Interactiveness	Communication Skill	Practical Knowledge	Job offer
1.	<9	No	Moderate	Good	Yes
2.	≥9	No	Moderate	Avg	No
3.	≥9	No	Moderate	Avg	No
4.	≥9	Yes	Moderate	Good	Yes
5.	≥9	Yes	Moderate	Good	Yes.



Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
data = pd.read_csv('iris.csv')
```

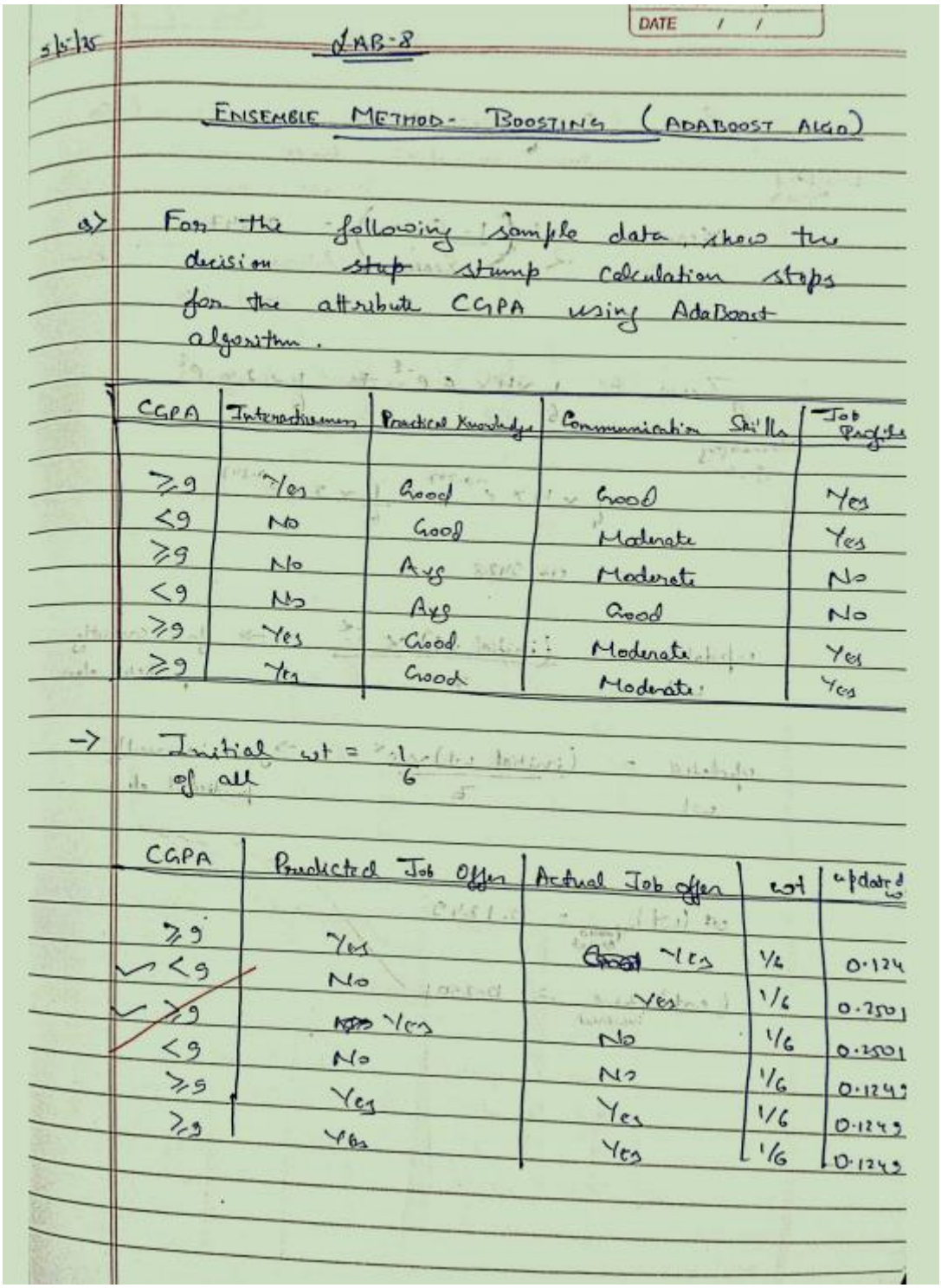
```
X = data.drop('species', axis=1)
y = data['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
rf_classifier = RandomForestClassifier(random_state=42)
rf_classifier.fit(X_train, y_train)
y_pred = rf_classifier.predict(X_test)
default_accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy with default n_estimators (10): {default_accuracy}")

best_n_estimators = 10
best_accuracy = default_accuracy
for n_estimators in range(1, 101):
    rf_classifier = RandomForestClassifier(n_estimators=n_estimators,
random_state=42)
    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"Best accuracy: {best_accuracy} achieved with n_estimators:
{best_n_estimators}")
from sklearn.metrics import precision_score, recall_score, f1_score
best_rf_classifier = RandomForestClassifier(n_estimators=best_n_estimators,
random_state=42)
```

```
best_rf_classifier.fit(X_train, y_train)

best_y_pred = best_rf_classifier.predict(X_test)

precision = precision_score(y_test, best_y_pred, average='weighted')

recall = recall_score(y_test, best_y_pred, average='weighted')

f1 = f1_score(y_test, best_y_pred, average='weighted')

print(f"Precision: {precision}")

print(f"Recall: {recall}")

print(f"F1-score: {f1}")

from sklearn.metrics import confusion_matrix

import seaborn as sns

import matplotlib.pyplot as plt

cm = confusion_matrix(y_test, best_y_pred)

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',

    xticklabels=data['species'].unique(),

    yticklabels=data['species'].unique())

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()
```

Program 9

Implement Boosting ensemble method on a given dataset.

Screenshot:



5/5/25 LAB-8

DATE / /

ENSEMBLE METHOD - BOOSTING (ADABOOST ALGO)

Q) For the following sample data, show the decision ~~step~~ stump calculation steps for the attribute CGPA using AdaBoost algorithm.

CGPA	Interactiveness	Practical Knowledge	Communication Skills	Job Profile
≥9	Yes	Good	Good	Yes
<9	No	Good	Moderate	Yes
≥9	No	Avg	Moderate	No
<9	No	Avg	Good	No
≥9	Yes	Good	Moderate	Yes
≥9	Yes	Good	Moderate	Yes

→ Initial wt = 1/6 of all

CGPA	Predicted Job Offer	Actual Job offer	wt	updated wt
≥9	Yes	~~Good~~ Yes	1/6	0.124
<9	No	Yes	1/6	0.2501
≥9	~~No~~ Yes	No	1/6	0.2501
<9	No	No	1/6	0.1249
≥9	Yes	Yes	1/6	0.1249
≥9	Yes	Yes	1/6	0.1249

DATE

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

Weighted Error

$\alpha_{CGPA} = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_{CGPA}}{\varepsilon_{CGPA}}\right) = 0.347$

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

Normalizing factor

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

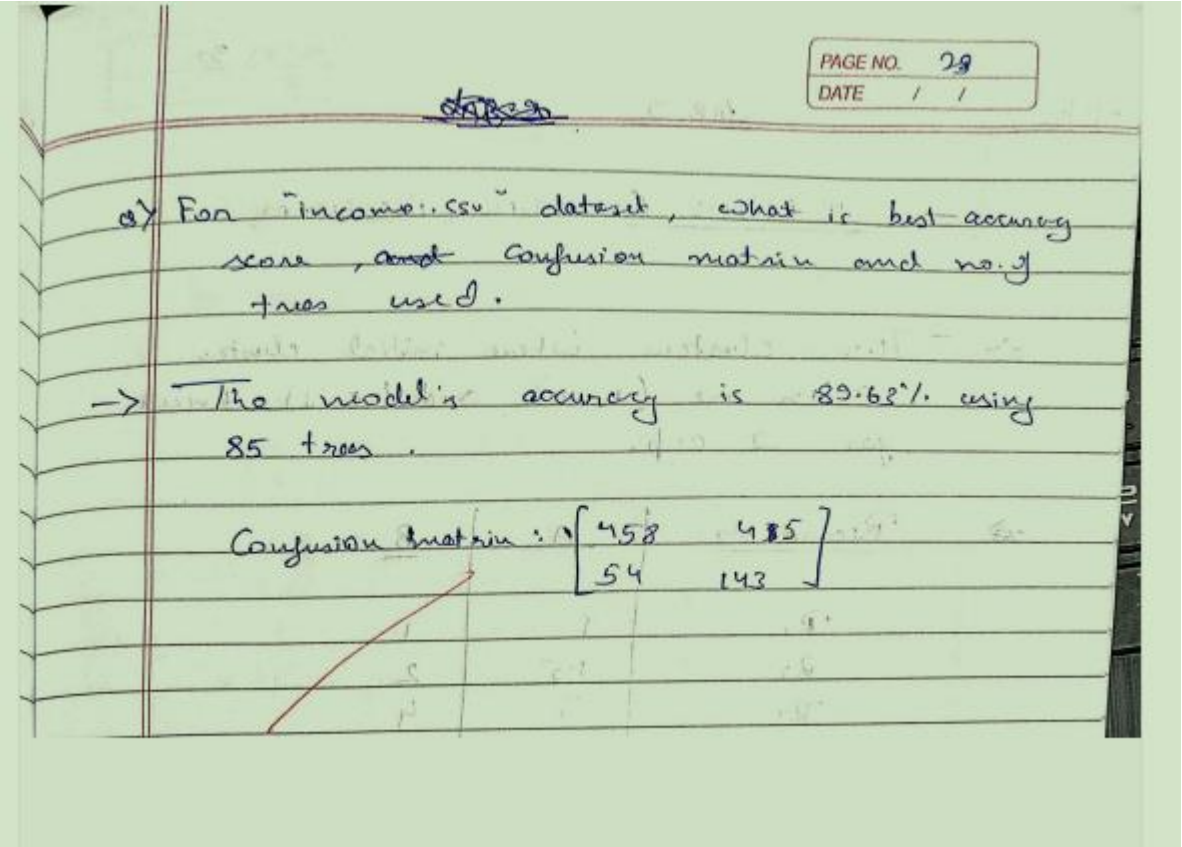
$= 0.9428$

updated wt $= \frac{(\text{initial wt}) \times e^{-\alpha}}{Z}$ → for correctly predicted class

updated wt $= \frac{(\text{initial wt}) \times e^{\alpha}}{Z}$ → for incorrectly predicted class

$(wt)_{\text{updated correct}} = 0.1249$

$(wt)_{\text{updated incorrect}} = 0.2501$



Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
df = pd.read_csv('income.csv')

label_encoders = {}
for column in df.columns:
    if df[column].dtype == 'object':
            le = LabelEncoder()
            df[column] = le.fit_transform(df[column])
```

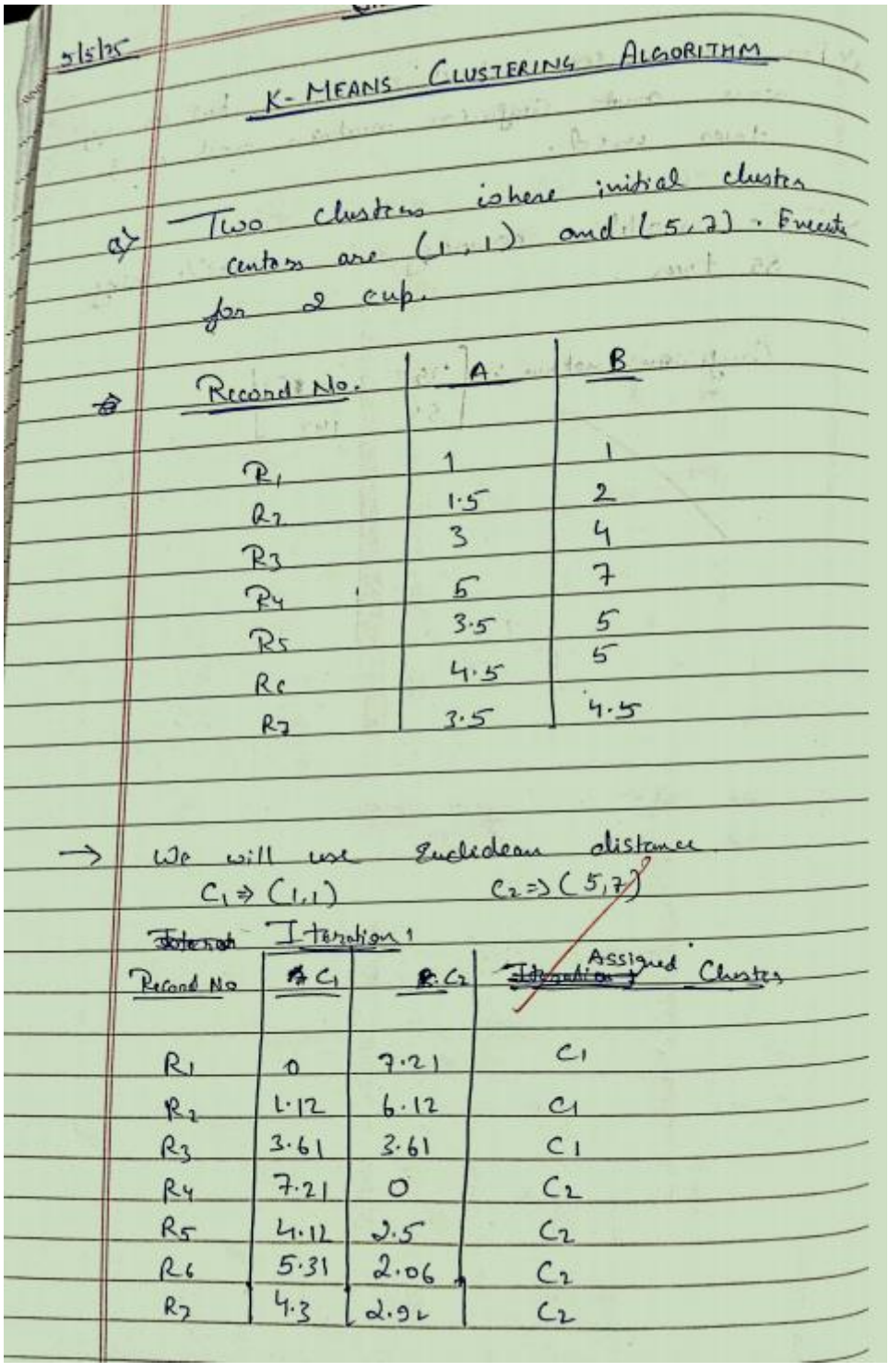
```
            label_encoders[column] = le

X = df.drop('income_level', axis=1)
y = df['income_level']

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

abc = AdaBoostClassifier(n_estimators=10, random_state=0)
abc.fit(X_train, y_train)
y_pred = abc.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy with n_estimators=10: {accuracy}")

best_accuracy = 0
best_n_estimators = 0
for n_estimators in range(1, 101):
    abc = AdaBoostClassifier(n_estimators=n_estimators, random_state=0)
    abc.fit(X_train, y_train)
    y_pred = abc.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}")
```

```
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt


abc = AdaBoostClassifier(n_estimators=73, random_state=0)
abc.fit(X_train, y_train)
y_pred = abc.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy with n_estimators=73: {accuracy}")
print(classification_report(y_test, y_pred))
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",
xticklabels=["<=50K", ">50K"], yticklabels=["<=50K", ">50K"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix Heatmap")
plt.show()
```

Program 10

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

Screenshot:



3/5/25

K-MEANS CLUSTERING ALGORITHM

Q) Two clusters i.e here initial clusters centers are (1,1) and (5,7). Execute for 2 epo.

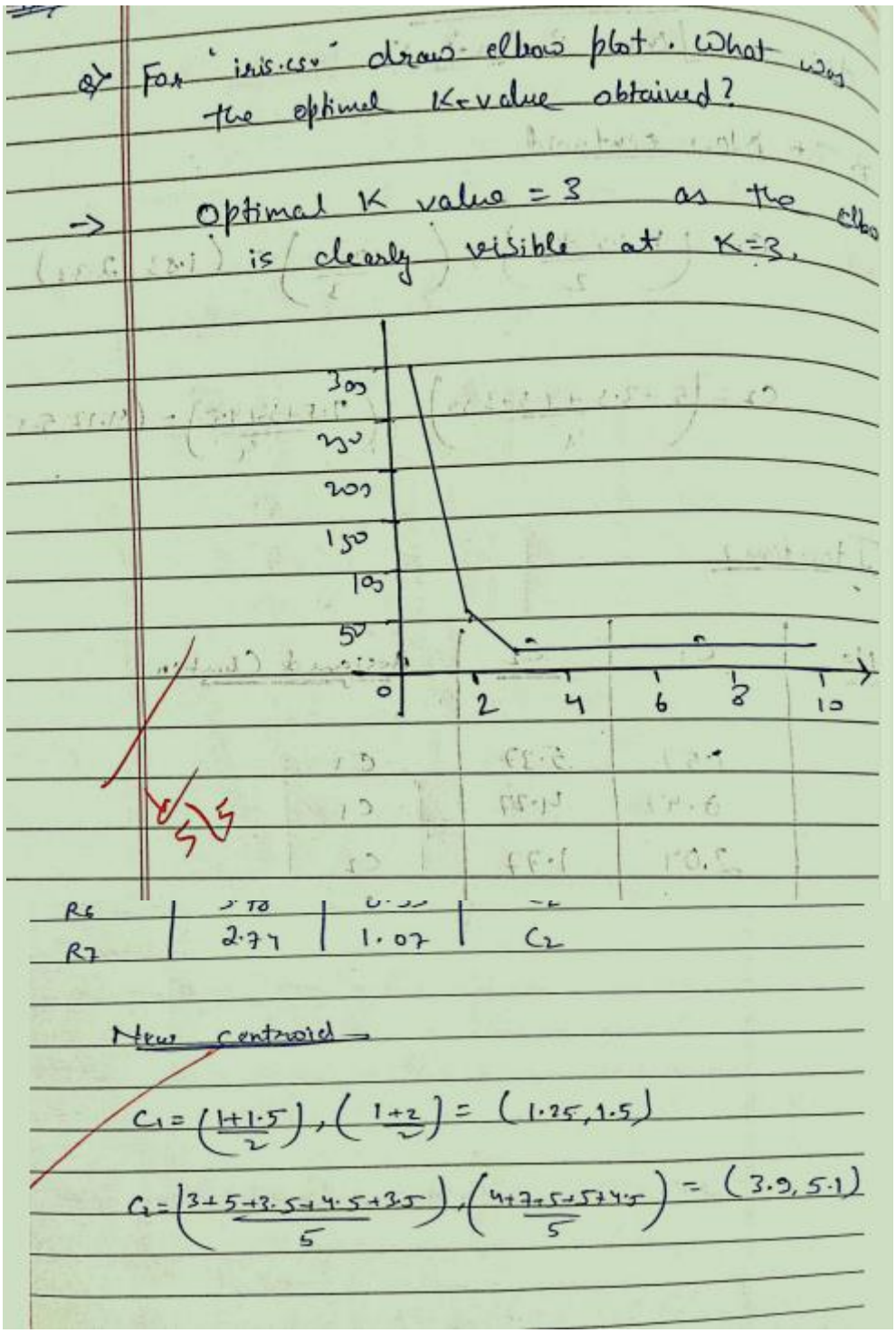
Record No.	A	B
R_1	1	1
R_2	1.5	2
R_3	3	4
R_4	5	7
R_5	3.5	5
R_6	4.5	5
R_7	3.5	4.5

→ We will use Euclidean distance.

$C_1 \Rightarrow (1,1)$ $C_2 \Rightarrow (5,7)$

Iteration 1

Record No	C_1	C_2	Assigned Clusters
R_1	0	7.21	C_1
R_2	1.12	6.12	C_1
R_3	3.61	3.61	C_1
R_4	7.21	0	C_2
R_5	4.12	2.5	C_2
R_6	5.31	2.06	C_2
R_7	4.3	2.92	C_2



Code:

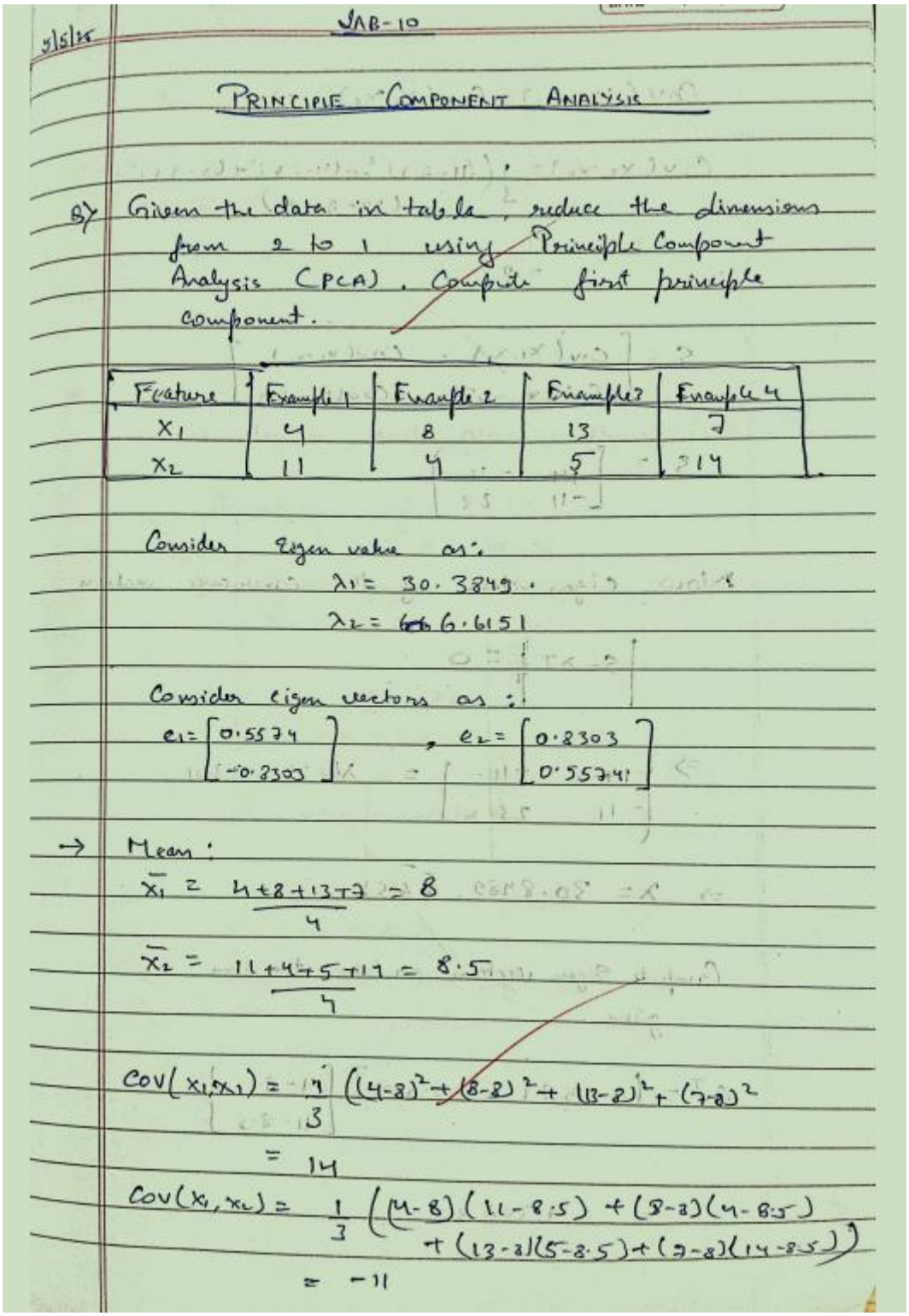
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

```
from sklearn.preprocessing import MinMaxScaler
df = pd.read_csv('iris.csv')
df = df[['petal_width', 'petal_length']]
scaler = MinMaxScaler()
df[['petal_width', 'petal_length']] = scaler.fit_transform(df[['petal_width',
'petal_length']])
sse = []
k_rng = range(1, 10)
for k in k_rng:
    km = KMeans(n_clusters=k)
    km.fit(df)
    sse.append(km.inertia_)

plt.xlabel('K')
plt.ylabel('Sum of squared error')
plt.plot(k_rng, sse)
plt.show()
kmeans = KMeans(n_clusters=3, random_state=0)
kmeans.fit(df)
df['cluster'] = kmeans.labels_
plt.scatter(df['petal_width'], df['petal_length'], c=df['cluster'],
cmap='viridis')
plt.xlabel('Petal Width')
plt.ylabel('Petal Length')
plt.title('K-Means Clustering of Iris Flowers')
plt.show()
```

Program 11

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

Screenshot:



LAB-10

PRINCIPLE COMPONENT ANALYSIS

Q) Given the data in table, reduce the dimensions from 2 to 1 using Principle Component Analysis (PCA). Compute first principle component.

Feature	Example 1	Example 2	Example 3	Example 4
X_1	4	8	13	7
X_2	11	4	5	14

Consider Eigen value as:

$\lambda_1 = 30.3849$

$\lambda_2 = 6.6151$

Consider eigen vectors as:

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

→ Mean:

$\bar{x_1} = \frac{4+8+13+7}{4} = 8$

$\bar{x_2} = \frac{11+4+5+14}{4} = 8.5$

$Cov(x_1, x_1) = \frac{1}{3}\left((4-8)^2 + (8-8)^2 + (13-8)^2 + (7-8)^2\right)$

$= 14$

$Cov(x_1, x_2) = \frac{1}{3}\left((4-8)(11-8.5) + (8-8)(4-8.5) + (13-8)(5-8.5) + (7-8)(14-8.5)\right)$

$= -11$

$Cov(x_1, x_2) = Cov(x_2, x_1) = -11$

$Cov(x_2, x_2) = \frac{1}{3}\left((11-8.5)^2 + (4-8.5)^2 + (5-8.5)^2 + (14-8.5)^2\right)$

$= 23$

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

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

Now, eigen values of the covariance matrix,

$|S - \lambda I| = 0$

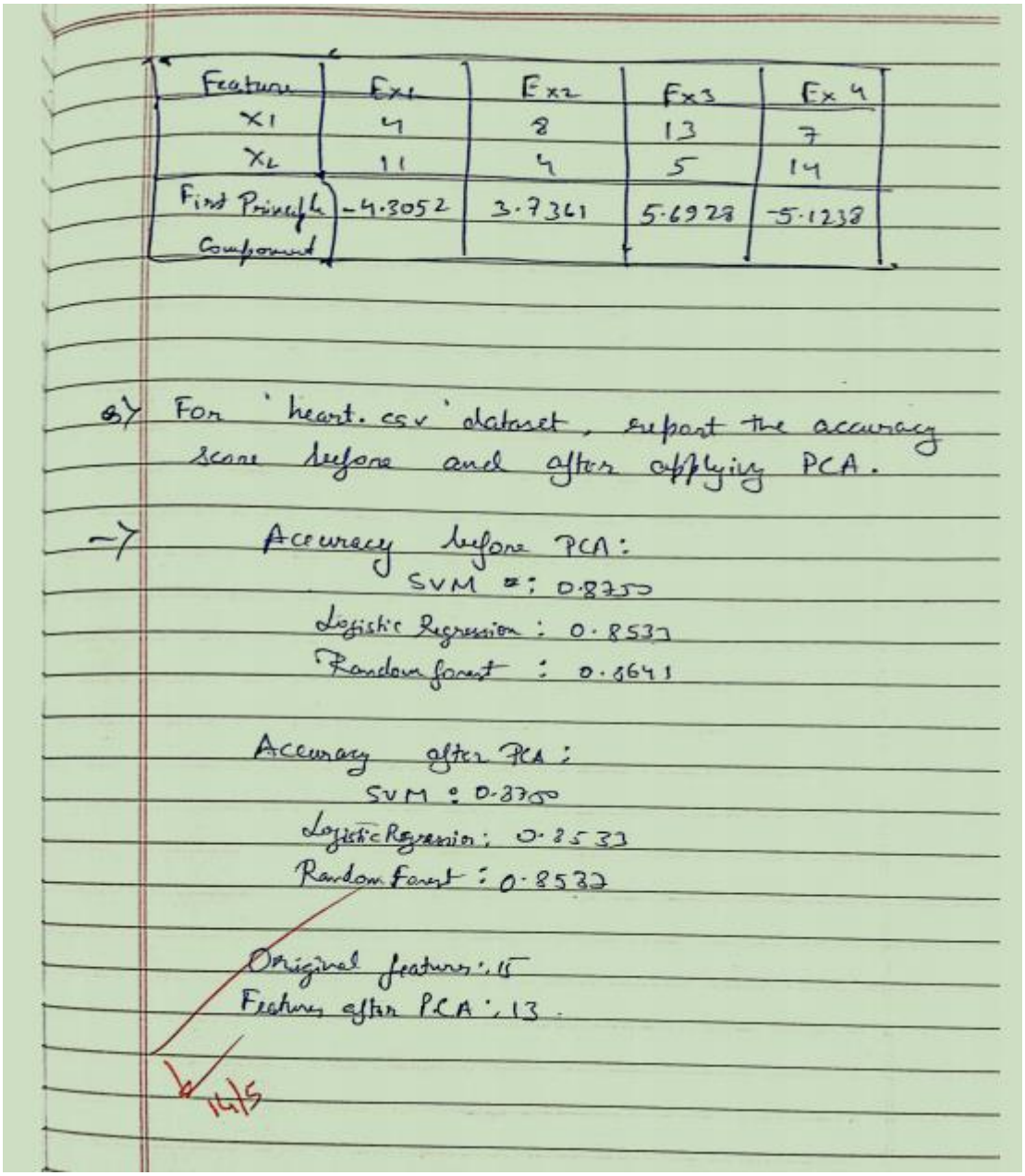
$\Rightarrow \begin{vmatrix} 14-\lambda & -11 \\ -11 & 23-\lambda \end{vmatrix} = \lambda^2 - 37\lambda + 201$

$\Rightarrow \lambda = 30.8489, 6.6151$

Compute Eigen vectors or take the ones given.

$e_1^T = \begin{bmatrix} 0.5574 & -0.8305 \end{bmatrix} \begin{bmatrix} 4-8 \\ 11-8.5 \end{bmatrix}$

$= -4.30535$



Feature	Ex1	Ex2	Ex3	Ex4
x1	4	8	13	7
x2	11	4	5	14
First Principle Component	-4.3052	3.7341	5.6928	-5.1238

8) For 'heart.csv' dataset, report the accuracy score before and after applying PCA.

→ Accuracy before PCA:
SVM : 0.8750
Logistic Regression : 0.8537
Random forest : 0.8641

Accuracy after PCA:
SVM : 0.8750
Logistic Regression : 0.8537
Random Forest : 0.8537

Original features : 15
Features after PCA : 13.

14/5

Code:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
df = pd.read_csv("heart.csv")
text_cols = df.select_dtypes(include=['object']).columns
le = LabelEncoder()
ohe = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
```

```
for col in text_cols:
    df[col + '_le'] = le.fit_transform(df[col])
    ohe_results = ohe.fit_transform(df[[col]])
    df_ohe = pd.DataFrame(ohe_results, columns=[f"{col}_{i}" for i in
range(ohe_results.shape[1])])
    df = pd.concat([df, df_ohe], axis = 1)
df = df.drop(text_cols, axis=1)
print(df.head())
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, MinMaxScaler
df = pd.read_csv("heart.csv")
text_cols = df.select_dtypes(include=['object']).columns
le = LabelEncoder()
ohe = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
for col in text_cols:
df[col + '_le'] = le.fit_transform(df[col])
ohe_results = ohe.fit_transform(df[[col]])
df_ohe = pd.DataFrame(ohe_results, columns=[f"{col}_{i}" for i in
range(ohe_results.shape[1])])
df = pd.concat([df, df_ohe], axis = 1)
df = df.drop(text_cols, axis=1)
scaler = MinMaxScaler()
scaled_values = scaler.fit_transform(df)
df_scaled = pd.DataFrame(scaled_values, columns=df.columns)
print(df_scaled.head())
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
X = df_scaled.drop('target', axis=1)
y = df_scaled['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
svm_model = SVC()
lr_model = LogisticRegression()
rf_model = RandomForestClassifier()
models = {'SVM': svm_model,
'Logistic Regression': lr_model,
'Random Forest': rf_model
}
results = {}
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    results[name] = accuracy
best_model = max(results, key=results.get)
best_accuracy = results[best_model]
print("Model Accuracies:")
for name, accuracy in results.items():
    print(f"{name}: {accuracy}")
print(f"\nBest Model: {best_model} with accuracy: {best_accuracy}")
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components=10)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)
results_pca = {}
for name, model in models.items():
    model.fit(X_train_pca, y_train)
    y_pred_pca = model.predict(X_test_pca)
    accuracy_pca = accuracy_score(y_test, y_pred_pca)
    results_pca[name] = accuracy_pca
best_model_pca = max(results_pca, key=results_pca.get)
best_accuracy_pca = results_pca[best_model_pca]
print("\nModel Accuracies after PCA:")
for name, accuracy in results_pca.items():
    print(f"{name}: {accuracy}")
print(f"\nBest Model after PCA: {best_model_pca} with accuracy:
{best_accuracy_pca}")
```