VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.



LAB REPORT

on

Machine Learning (23CS6PCMAL)

Submitted by

Pratik Jana(1BM22CS356)

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 **Pratik Jana (1BM22CS356)**, 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 a Machine Learning (23CS6PCMAL) work prescribed for the said degree.

Lab Faculty Incharge

Name: **Ms. Saritha A N**Assistant Professor

Department of CSE, BMSCE

Dr. Kavitha Sooda Professor & HOD

Department of CSE, BMSCE

Index

Sl. No.	Date	Experiment Title	Page No.
1	21-2-2025	Write a python program to import and export data using Pandas library functions	1-3
2	3-3-2025	Demonstrate various data pre-processing techniques for a given dataset	4-8
3	10-3-2025	Implement Linear and Multi-Linear Regression algorithm using appropriate dataset	9-15
4	17-3-2025	Build Logistic Regression Model for a given dataset	16-20
5	24-3-2025	Use an appropriate data set for building the decision tree (ID3) and apply this knowledge to classify a new sample	21-25
6	7-4-2025	Build KNN Classification model for a given dataset	26-28
7	5-5-2025	Implement Random Forest ensemble method on a given dataset	29-33
8	5-5-2025	Implement Boosting ensemble method on a given dataset	34-40
9	12-5-2025	Build k-Means algorithm to cluster a set of data stored in a .CSV file	41-44
10	12-5-2025	Implement Dimensionality reduction using Principal Component Analysis (PCA) method	45-49

GITHUB LINK: pratik03092003/ML-LAB

Program 1

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

Screenshot

1	Initialing values directly into Datapame	99.
	impart pandas as pd	
	data= {	
	'Name' : [A.B', c', D']	Mile I
	Martes': [10,9,8,10]	
	USN' : [351, 852, 351, 356]	
	S A Design and a source of the delivery of the second	0.61
· Lu	11 = pd-datatrame (dafa)	13
	aj= pd-datajrame (dafa) print ('Sampledata')	
(7)	print (af head ())	
170	MORE DESCRIPTION OF THE PROPERTY OF THE PARTY OF THE PART	(1
m	to divine the point of the day State of the	gam(kg
2-	Importing dataset from sklearn dataset from sklearn dataset import load diabetes	Mater
	d= pd data frame (diabetisdata, edeems=d	iabela
	feature names	
	di (target) = diabelts target	
	print (af head ()	
3-	importing dataset from specific csv file .	
	from skleam dataset import wad diabet	g
	file = pa. read_csv (file_path)	
	punt ('sampledata')	- The Paris
	print (of head !)	A STATE OF THE STA
	Ashir Traditional Company of the Com	
	A THE PARTY OF THE	Bonn
	THE RESIDENCE OF THE PARTY OF T	

```
Code:
import pandas as pd
from sklearn.datasets import load_diabetes
# Part 1: Student data
student_data = {
  'USN': ['1A18CS001', '1A18CS002', '1A18CS003', '1A18CS004', '1A18CS005'],
  'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
  'Marks': [85, 78, 92, 88, 75]
}
student_df = pd.DataFrame(student_data)
print("Student Data:")
print(student_df)
print("\n" + "-"*50 + "\n")
# Part 2: Load diabetes dataset from sklearn
diabetes_data = load_diabetes()
diabetes_df = pd.DataFrame(diabetes_data.data, columns=diabetes_data.feature_names)
diabetes_df['target'] = diabetes_data.target
print("Scikit-learn Diabetes Dataset:")
print(diabetes_df.head())
print("\n" + "-"*50 + "\n")
```

Part 3: Load sample sales data from CSV

```
try:
  sales_df = pd.read_csv('sample_sales_data.csv')
  print("Sample Sales Data:")
  print(sales_df.head())
except FileNotFoundError:
  print("sample_sales_data.csv not found.")
print("\n" + "-"*50 + "\n")
# Part 4: Load diabetes dataset from external CSV
try:
  diabetes_csv_df = pd.read_csv('Dataset of Diabetes .csv')
  print("Diabetes Dataset from CSV:")
  print(diabetes_csv_df.head())
except FileNotFoundError:
  print("Dataset of Diabetes .csv not found.")
```

Program 2

Demonstrate various data pre-processing techniques for a given dataset Screenshot:

	DATE:
4.	Download dataset from existing dataset like Kaggle
	Ju-path= /content / Dataset of diabetes CSV' ay-pa-read-isv (file-path)
	print ('Sample dala:) print (dj. head ())
5.	using the code given; do the excercix of stock market analysis, consider the following.
1.4	distance in Sugar
D	Mahindra Bank Ud.
-lolet	HICKURS = ["HDFC BANK : NS" , " ICICI Bank"; KOFAK BANK : NS"]
	Slantoale: 2024-01-01 enadate: 2024-12-30
	plot the coding price and daily return.
	import yfinamo as yi (basel 10) but
-6	import matplotub puplot as ptt
	Lickers = ['HOF CBANK NS", 'ICICIBANY'S
	aata= gf. download (tickets, Start = 2014-01-01) end - 2024-12-30, groupby-tickel)
	ousing-price - pd-dataframe ()

=	
1111	daily returns - pd. data-Frame ()
4	Low tichet in tichen:
	of the state of th
	- Land The DU 13 MOVE MARK INCOME
	daily-richirms [tickers] = stock data ["(wse"]
	get-change ()
	10-10-901
	955524-00-0-23
	Write the pymon code, consider the filename as
	White the pyrantione revision
fes	"housing.csv"
10000000	toward csv fute unto the DF
(ii)	display info. of all numerical
(ii)	to display, statistical into of all numerical
(tv)	to display the count of unique labels 'ocean proximity' to display which attribut values count greater
(v)	to display which attribut values count greats
	man 2170
	septy Paris / Abusti Author
	import pandas as pd af = pd-read_csv ('housing.csv") point (af-Info())
	al = pd-read_csv ('housing.csv')
	mint (al-Info())
	print (af. describe ())
	plast 4 state 1
	from skleain preprocessing import min Max Scalar
	, Slandard scalar
100	diabelts of = pd. read_(SV (*/coulon / pattern/
	Dafaset CSV")
	adult income of = pd. read-csv ("adul.qv")
	adult incine ag = pa racio = cor (
	le solum malle dishilts al-silect atunes (
	for column and a drabults - of-select - dtyres (
	diables of [inlumn], errors' were'
	adapte of (Munut)) Entris were

		· free
- 3		
	PAGE NO : DATE :	3
	avabeles-af-mean()	
	tunant of towns	pplace
11	steek data : data f irenia	mul
100	Q1 = af (colum) anahne (0.25). Q3 = af (colum) quante (0.76)	/
F 807	Que af Colum Janahne (0-25)	1
The second	130 ay (colum) quante (0.78)	\
	IOR - Q3-Q1	_
	LB = 91-1-5×19R	
	UB . 93+1.5 × IQR	
	All	
	Paradiological Paradiological	Ĭ)
	distance of our manifold.	1
'phones	to display the court of second laws bearings	1
udos	to despite which although palme (and a)	U)
	of taken to the transfer of the second	1
	Cours work	
	Consort paness or plant to	The second
	The benefit of the many can	_
	(Stelent-les) inter	_
-	(O strong (R) topo	_
Planty	Sid num totaline pours squery ministra mai	
	Sandard Soylak	
	Marine Van Supply Days he make a few might be	/

```
Code:
import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt
tickers = ["HDFCBANK.NS", "ICICIBANK.NS", "KOTAKBANK.NS"]
# Fetch historical data for the last 1 year
data = yf.download(tickers, start="2024-01-01", end="2024-12-30", group_by='ticker')
hdfc_data = data['HDFCBANK.NS']
hdfc_data['Daily Return'] = hdfc_data['Close'].pct_change()
ic_data = data['ICICIBANK.NS']
ic_data['Daily Return'] = ic_data['Close'].pct_change()
kb_data = data['KOTAKBANK.NS']
kb_data['Daily Return'] = kb_data['Close'].pct_change()
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
hdfc_data['Close'].plot(title="HDFC Bank - Closing Price")
plt.subplot(2, 1, 2)
hdfc_data['Daily Return'].plot(title="HDFC Bank - Daily Returns", color='orange')
```

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

<u>Program 3</u>
Implement Linear and Multi-Linear Regression algorithm using appropriate dataset

Screensh	hot	
alala	PAGE NO: DATE:	1
9/8/25	Tags	
	Please execute Linear regression:-	
-	(-) ()(-)	
	$\alpha_1 = (\overline{xy}) - (\overline{x})(\overline{y}) / (x^2) - (\overline{x})^2$	
	Gop =	
	$a_0 = (\overline{y}) - a_1 \times \overline{x}$	
	Lan Distance	
	xº yi	
	1 12 Agreed to the little	
	2 18	
	3 22	
	4 28 1119 129 129 129 10	1
Yelfal	Man Pelalist Colonia	
	X=np.array ([1,2,3,4])	-
	X= np. array ([1,2,3,4]) y= np. array ([2,4,5,9])	
	import numpy as np	
	import numpy as np import matphothib pyphot as plt	
		- 9
-	def linear-regression (x, y):	
	def linear-regression (X14): n=lin(X)	
	Y-mean = np. mean (x) Y-mean = np. mean (y)	
		200
	b1 = np. sum ((x-x-mean) + (y-y-mean)	16
	b = np. sum ((x-x-mean)+ (y-y-mean) mp. sum (x-x-mean) +2	1
	bo= y-mean - b1 + x-mean	
	return 60,61	
	79 =	
1000		

	PAGE NO:
	DATE:
	do sa patempa demons
	bo, bs = Unear_regression (xi, yi) print (f" y= { bo: . 2f } +(b1: . 2f 3x")
	bo, bi = waw sagaranty sign
	mnt (1 "9= 1 Db: . 2+) 1(0.1.20)
U	
	ptt. scatta (xi)yi, color = blue, label: Data por ptt. plot(xi, bo+b) +xi, color='red', label='Best-fit-l
	plt- plot (XI, DO+b) -XI, COLOT= 180 , (about - DO) file
	plt. plot (X, D8+0) X) cour as) land plt. xlabel ('X (Independent Variable)') plt. ylabel ('V (Dependent Variable)')
	Det ylabel (V (Dependent Variable))
	1 1 1 - 1 ()
	but Hitle (linea regusión)
	pt. Httl. (linea. Hegusion") pu-show ()
	The same of the sa
	Y= 6.20+5.60X
	(idiod) Anno
The same	The same of the state of the st
(3/40/3)	The state of the s
	1 - To Charles the Charles the
-	(Comment and complete trail) a lade land to the
	Made and bake again [] Wall to be fulled
	CHOSEAN DEATHERS NO CONTRACTOR OF THE CONTRACTOR
	Compressed the second state of the second stat
-	The Chapter and the second of the Chapter of
	Wall date U
	100110001
	The second secon
59-61 E-16-	

The state of the s	Matrix method PAGE NO: DATE:
	import numby as no
1	o in a model of librariat as plt.
1	x= np. column_sack ((np.ones (un(A)),x))
1	v= po column sack (inp. ones (in(A), x))
-	Y= np. anay (y). rushape (-1,1)
1	The second of the second of the second
1	B= rp-linalg · inv (X·T@ X) @ X·T@ Y
1	return B. Hatten ()
1	O hope I de
1	x: = phoneau ([1,2,3,4,5])
-	ni = nparray ([1,2,3,4,5]) yi = mp. array ([2,3,5,6,8])
-	of 110 1100 00 (22) 21 21 21 20 2
-	bo, b1 = Linear-regression-matrix (21,41)
	print (boj bi)
	prist (9-151)
	pll-scatter (21, y; color= blue , label & Data por
	plt. plot (xi, bo+b1 +xi, color-ired, laba=
	Best fithing)
	pll-xlabel ("x [Indyunder Variable)")
	pll-ylabel ("y (Independent Variable)")
	ple-liginal)
	pet title ("linear Reguession (Matrix Method)")
	pu-show()
	U-MEAN)
	4=0.30 + 1.50x
2	
1	
10	No.

Code:

import matplotlib.pyplot as plt

```
x = list(map(float, input("Enter X values separated by space: ").split()))
y = list(map(float, input("Enter Y values separated by space: ").split()))
n = len(x)
sum_x = sum(x)
sum_y = sum(y)
sum_x^2 = sum(xi^{**}2 \text{ for xi in x})
sum_xy = sum(x[i] * y[i] for i in range(n))
a1 = (n * sum_x - sum_x * sum_y) / (n * sum_x - sum_x * 2)
a0 = (sum_y - a1 * sum_x) / n
print("\nLinear Regression Equation:")
print(f''y = \{a0:.2f\} + \{a1:.2f\}x'')
x_test = float(input("\nEnter a value of x to predict y: "))
y_pred = a0 + a1 * x_test
print(f"Predicted y: {y_pred:.2f}")
plt.scatter(x, y, color='blue', label='Data Points')
x_{line} = [min(x) - 1, max(x) + 1]
y_{line} = [a0 + a1 * x_val for x_val in x_line]
plt.plot(x_line, y_line, color='red', label='Regression Line')
plt.scatter(x_test, y_pred, color='green', label=f'Prediction: ({x_test}, {y_pred:.2f})')
plt.xlabel('X values')
plt.ylabel('Y values')
```

```
plt.title('Linear Regression Visualization')
plt.legend()
plt.grid(True)
plt.show()
import numpy as np
import matplotlib.pyplot as plt
x = np.array(list(map(float, input("Enter X values separated by space: ").split())))
y = np.array(list(map(float, input("Enter Y values separated by space: ").split())))
X = np.c_[np.ones(len(x)), x]
beta = np.linalg.inv(X.T @ X) @ X.T @ y
print("\nLinear Regression Equation (Matrix Form):")
print(f''y = \{beta[0]:.2f\} + \{beta[1]:.2f\}x'')
x_test = float(input("\nEnter a value of x to predict y: "))
y_pred = beta[0] + beta[1] * x_test
print(f"Predicted y: {y_pred:.2f}")
plt.scatter(x, y, color='blue', label='Data Points')
x_{line} = np.linspace(min(x) - 1, max(x) + 1, 100)
y_{line} = beta[0] + beta[1] * x_{line}
plt.plot(x_line, y_line, color='red', label='Regression Line')
plt.scatter(x_test, y_pred, color='green', label=f'Prediction: ({x_test}, {y_pred:.2f})')
plt.xlabel('X values')
plt.ylabel('Y values')
```

```
plt.title('Linear Regression (Matrix Form)')
plt.legend()
plt.grid(True)
plt.show()
```

Program 4

Build Logistic Regression Model for a given dataset

Screenshot

4	tare:
	Logistic regusion
(1) a)	$40) = \frac{1}{1 + e^{-(x_0 + a_1 x)}}$
4)	probability for 7 hours
	yor x = 7 umpart numpy as mp
	00,04 = -510.8
	preto 1/(1+np.exp(-(ao +arz))) print (proh)
()	1° +1 day (Thursd = 0.5)
	predicted class= 'pass of proby= v o
	print (predicted class)
à(1)	Soft max: Ab away exp (7)/ np. sum (np. exp(2))
	mont (softmax)
	TOTAL STREET STREET STREET
	AND THE REAL PROPERTY OF THE PARTY OF THE PARTY.

	110 7 000
	HR comma_SV. CSV
1)	Solisfaction Level (-0.39 correlation with Life)
	-> Employees with low satisfactory are
	more likely to leave
a)	Number of projects (0.14)
	-> toployer with 60 few or to many
	proxices tends to leave
3)	monthy dug nous (0.13 couleliss)
-/	10 1 110
	for longlime wilhout promotion tool to
	leave
-	Yan
	200
	1) pata preprouning steps - feature selection
	- feature scaling
-	- Frain lost split
_[[0]]	- class incoding
	2) Null values are identified using:
	700 dala · unull () · sum ()
-	3) Most frequently mas identified does 1) Reptider (3) & Amphibians (5)
	1) Replace (3) & Amphibians (5)
	SIMULATION SIMILAR
	3) Juh (4) & Amphibans (5)
1	
- 2	Aller .

```
Code:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
# Load dataset
df = pd.read_csv("HR_comma_sep.csv")
# Convert categorical variables to numerical
label_enc = LabelEncoder()
```

```
df["salary"] = label_enc.fit_transform(df["salary"])
df["Department"] = label_enc.fit_transform(df["Department"])
# Step 1: Exploratory Data Analysis (EDA)
correlation = df.corr()["left"].sort_values(ascending=False)
print("\nFeature Correlation with Employee Retention:\n", correlation)
# Step 2: Impact of Salary on Retention
plt.figure(figsize=(6,4))
sns.barplot(x="salary", y="left", data=df, ci=None)
plt.xlabel("Salary Level (Encoded)")
plt.ylabel("Retention Rate")
plt.title("Impact of Salary on Employee Retention")
plt.show()
# Step 3: Correlation between Department and Retention
plt.figure(figsize=(8,4))
sns.barplot(x="Department", y="left", data=df, ci=None)
plt.xlabel("Department (Encoded)")
plt.ylabel("Retention Rate")
plt.title("Department vs Employee Retention")
plt.show()
```

Step 4: Logistic Regression Model

```
features = ["satisfaction_level", "last_evaluation", "number_project", "average_montly_hours",
"time_spend_company", "salary"]
X = df[features]
y = 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)
# Step 5: Model Accuracy
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"\nModel Accuracy: {accuracy:.2f}")
# Display Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:\n", conf_matrix)
# Display Classification Report
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

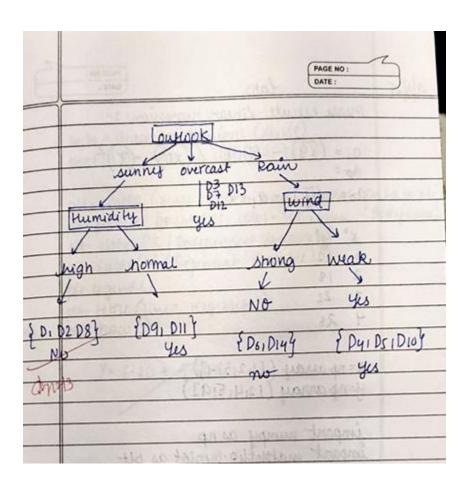
Program 5

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

Screenshot

	PAGE NO:
3/25	GTGTGTGTGTGTGT
4	
	ID3 algorithm for making descision tree
	1 40 (1 6460) 300 00 30 30 40
	5- horu(s) = 5 p. log D
	Entropy (s) = $\sum_{i=1}^{\infty} p_i \log_2 p_i$
	TRILLIAND TapesTetch month
	(- Courtage)
	Grain (Southook) = Enhopy (S) - [ISV] Enh
	= Enlipt & c2 El 181 (3
	ve(R)
	import pandas as pa
	uniport numby as up
	1 (Francisco And The Science of the Control of the
Tu	def entropy (data-col):
24	and entropy (data-col) = np. unique (data-col, unique classes, counts = np. unique (data-col,
	I have been a supplied to the
	prob- count count sum()
	return -np. sum (prot " np. log2 (prob))
	TA TO ANY
	(med) amore blade on san
	1. 1. De voe landend land
	def gain (difit):
	enhopy= enhony (a[t]).
	featur-val geount- unique (d[f])
	weighted_entropy=0
	for I in trange len (feature val) }. Subset = a[a[f]] == feature-val [?]
-	subut entopy- entropy (Subset [t])
-	wreghted enboyingt = (count(i)/(un(data)).
	HELLAN CHARDY - WEIGHT ENTONY Substantony
100	The state of the s

	42:52	PAGE NO : DATE :	
	def at (artif)	Canada	
arl.	ALLEND LANGER OF THE CONTRACTOR	IDS OF	
	if hen (np. unique (data I	[t]) == (*	
return np. unique (data [
3130	if lin(features) ==0		
	return data [target].	mode () [0]	
11.03	best-featur= max (f, key = gain (a	-lambda feat:	
121	gan (o	(, F , +))	
	at = f. best-feature: 5 37	NAMES AND ASSESSED.	
	CAN BE WOOD	WALLY ALACTICAL	
	for v. in np. unique (a [best_ Subset = data [data [best	feature]);	
	Subset = data [data [best	== val	
Jay.	des crision tree [best-feature][vai] = at (subse		
MCE	target, run-paperes)		
		so select	
-	return desission tree		
66	Charles and Charles and Charles and	- marker -	
	af- pd. data Frame (Data	rec·(SV")	
	af- pd. data Frame (Data)	
	final tree = at (daysut	Make assess (
	target - blay".	* stateday	
	fratures . I'we mu	Stratuted.	
	'kmp'	hatdown -	
	Yumi as	ly',	
	fatures = I' we mue 'temp' 'tumi cui 'tuna'	17	
	print (final bre)		
63	and the state of	ROTTLES.	
. 3	A day to the Arriva III		



```
Code:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
# Load dataset
df = pd.read_csv("HR_comma_sep.csv")
# Convert categorical variables to numerical
label_enc = LabelEncoder()
df["salary"] = label_enc.fit_transform(df["salary"])
df["Department"] = label_enc.fit_transform(df["Department"])
# Step 1: Exploratory Data Analysis (EDA)
correlation = df.corr()["left"].sort_values(ascending=False)
print("\nFeature Correlation with Employee Retention:\n", correlation)
# Step 2: Impact of Salary on Retention
plt.figure(figsize=(6,4))
```

```
sns.barplot(x="salary", y="left", data=df, ci=None)
plt.xlabel("Salary Level (Encoded)")
plt.ylabel("Retention Rate")
plt.title("Impact of Salary on Employee Retention")
plt.show()
# Step 3: Correlation between Department and Retention
plt.figure(figsize=(8,4))
sns.barplot(x="Department", y="left", data=df, ci=None)
plt.xlabel("Department (Encoded)")
plt.ylabel("Retention Rate")
plt.title("Department vs Employee Retention")
plt.show()
# Step 4: Logistic Regression Model
features = ["satisfaction_level", "last_evaluation", "number_project", "average_montly_hours",
"time_spend_company", "salary"]
X = df[features]
y = 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)
```

```
# Step 5: Model Accuracy
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"\nModel Accuracy: {accuracy:.2f}")

# Display Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:\n", conf_matrix)

# Display Classification Report
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

Program 6

Build KNN Classification model for a given dataset

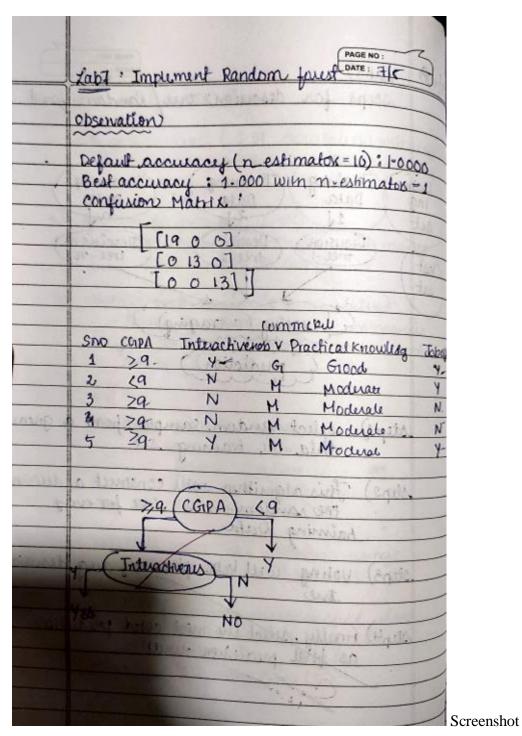
Screenshot

			DATE: 1
Zat	10 KA	IN .	
- DISH	(11)	(214)	10 4011
1960	00.	V52	- eı
1.512	V1.25	V2125	CI
314	V13	V13 000	Ce
1817	VE		C2
345	V2715	16.52	C2
4518	131.12	MILE	Cz
35141	118.8	18.5	CZ
	1.31.1	7. 10000	· University
No.	44,44	+2 - (1.2	(2:1:2)
C2 -	. (391	2-1)	
	(35.15)	(12,00	clusta
(111)	0.56	3.55	CI
920 (2151)		4.04	02
(314) 250		1-42	Cg.
(1D) 6:10		2.16	C2
(35, 5)	2.85	0.41	CL
1455) 320		0.6)	Ca
36140)	12.63	0.21	ce
chul	(1)	(3510) (35)	

```
Code:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
file_path = input("Enter the path to your CSV file: ")
df = pd.read_csv(file_path)
# Automatically select the target column (assuming it's the last column)
target_column = df.columns[-1]
# Splitting features (X) and target variable (y)
X = df.iloc[:, :-1] # All columns except last (features)
y = df.iloc[:, -1] # Last column (target)
# Encode categorical target variable if necessary
if v.dtvpe == 'object':
  label_encoder = LabelEncoder()
  y = label_encoder.fit_transform(y)
# Convert categorical features to numerical if any
X = pd.get dummies(X, drop first=True)
# Split the dataset (80% training, 20% testing)
X train, X test, y train, y test = train test split(X, y, test size=0.2, random state=42)
# Standardize the data
scaler = StandardScaler()
X train = scaler.fit transform(X train)
X_{\text{test}} = \text{scaler.transform}(X_{\text{test}})
# Determine optimal k (square root heuristic)
k = int(np.sqrt(len(y_train)))
if k % 2 == 0:
  k += 1
# Train the KNN classifier
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)
# Make predictions
y_pred = knn.predict(X_test)
```

```
# Display accuracy score
accuracy = accuracy_score(y_test, y_pred)
print(f"\nAccuracy Score: {accuracy:.2f}")
# Display confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:\n", conf_matrix)
# Display classification report
class_report = classification_report(y_test, y_pred)
print("\nClassification Report:\n", class_report)
#Plot Confusion Matrix
plt.figure(figsize=(6,5))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=np.unique(y),
yticklabels=np.unique(y))
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```

Program 7
Implement Random Forest ensemble method on a given dataset



2100	NAM		PAGE DATE	Medical Information of the Control o
	(Iv	tuachiens	Jyes	374
	NO		Jyes	
-		The dree	yes	010
-	Knowled	AVG	72293	
900	adula in	Visuality	ASE	rant
Creation	D. 1.	Aurage)	DIC /21	- 6
1 2		Ne	19 50	0
ye	0 11	NON	DE ON	
1	N	i.t	Po nid	-
41	The V	12	P & 605	3
-	VY	V	BY COP	
1	THE PERSON NAMED IN	HADE		
-	March 1	37.7.1	1103	
-	Stars L	OBIN.	1 -0	1 4
Sanka.	Till hards	1 (3-1)	10 1 15 15 15 15 15 15 15 15 15 15 15 15 1	1
34.	1311	7 100 N		
No	2.04			
-		V	2000	
		×-9	1414	4.5
	1:41			
	(etch	· V o v lo	purp lann) :	51
o Attitude	(pr) 1 , 40	WA A STATE	*	
30003	(11500	H	The same	

Code: import pandas as pd from sklearn.ensemble import RandomForestClassifier from sklearn.model_selection import train_test_split from sklearn.metrics import accuracy_score import matplotlib.pyplot as plt # Load dataset df = pd.read_csv("iris.csv") # Features and target X = df.drop(columns=["species"]) y = df["species"] # Split into train/test # 1. Train with default n_estimators = 10 rf_default = RandomForestClassifier(n_estimators=10, random_state=42)

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
rf_default.fit(X_train, y_train)
y_pred_default = rf_default.predict(X_test)
default_score = accuracy_score(y_test, y_pred_default)
print(f'Accuracy with default (10 trees): {default_score:.4f}")
```

```
#2. Fine-tune n_estimators
scores = []
tree\_range = range(1, 51)
for n in tree_range:
  model = RandomForestClassifier(n_estimators=n, random_state=42)
  model.fit(X_train, y_train)
  y_pred = model.predict(X_test)
  acc = accuracy_score(y_test, y_pred)
  scores.append(acc)
# Plotting results
plt.figure(figsize=(10,5))
plt.plot(tree_range, scores, marker='o')
plt.title("Random Forest Accuracy vs Number of Trees")
plt.xlabel("Number of Trees (n_estimators)")
plt.ylabel("Accuracy")
plt.grid(True)
plt.show()
# Best result
best\_score = max(scores)
best_n = tree_range[scores.index(best_score)]
print(f'Best Accuracy = {best_score:.4f} with n_estimators = {best_n}")
```

Program 8

Implement Boosting ensemble method on a given dataset Screenshot

tabs				PAGE NO:
Adu	aboast a	lgorutm .	(42)	
٠٠			J.	
acc	sion) sh	ump for	attribute	Capa
Show.	CGPA	Achiel	Riedic	1
1	2017×19			tus a
2	MB (9	No	Y	×
3	No 79	NOW	N	
4 5	No <9	N	N	V
5	400 79	4		v. 1
6	4607/9	4		V
	E=1/6	THE STATE OF		
		(1-E)	1 en (5	16 = 10
				0.804
	6+01	e-0		
	2-234		.444	
• 2	= Good con	id x e-x *	nost +(weigh)	no of the
			100000	
	Fresh	447 x1/6)	+ 1x	2-254 x 4
	- 0.14	25 + 4	0-3123	

	11:			PAGE NO: DATE:
			AMM B	abst
	New- old	Zwxe-d	New	s(
	0+	2(-1)		ovr -12.0
	correct			
	19.	1/6 x 0.447	1/61	0.294
	10	7	2501	7 '
	0-1	10013	0.5	004
	(2	in the same	VEV	110
	63	12:01	25.5 0	2118
	(3)	Timil	7000	8 124
	observation	N 2.5 1	200	19486
	Default o	ccuacy = (3-8277	3100
	Default o	confusion M	ahu:	
	(Ell	722 387]	7-11	= 00
	12	138 1406] 1 3	
/	moing	100 trees (10 in each	batehv
	מנוגות,	(12.00)	and	
	13	71.9	0.56	(1,1)
	50	15/3/44	9:0	(3,20)
	0)	1.42	038.	(2,5)
	30	2116	01.0	(1.2)
	30	Tho	288	(202)
	(2)	(9.0)	306	(3)74)
	63	170	1253	

```
Code:
import pandas as pd
import numpy as np
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
import matplotlib.pyplot as plt
# Load the dataset
data = pd.read_csv('income.csv')
# Explore the dataset
print(data.head())
print("\nDataset info:")
print(data.info())
print("\nClass distribution:")
print(data['income_level'].value_counts())
# Split into features and target
X = data.drop('income\_level', axis=1)
y = data['income_level']
# Encode categorical features (one-hot encoding)
```

```
X = pd.get\_dummies(X)
# Check for missing values
if X.isnull().sum().sum() > 0:
  print("Missing values found. Filling missing values with column mean.")
  X = X.fillna(X.mean())
# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
  X, y, test_size=0.3, random_state=42, stratify=y)
# Define base estimator
base_estimator = DecisionTreeClassifier(max_depth=1) # Stump tree for AdaBoost
# Initial AdaBoost model with 10 estimators
ada_model = AdaBoostClassifier(
  estimator=base_estimator,
  n_estimators=10,
  random_state=42
)
# Train the model
ada_model.fit(X_train, y_train)
```

```
# Make predictions
y_pred = ada_model.predict(X_test)
# Evaluate initial model
print("\nInitial Model with 10 estimators:")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
# Fine-tune the number of trees
n_{estimators\_range} = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200]
train_scores = []
test_scores = []
for n in n_estimators_range:
  model = AdaBoostClassifier(
     estimator=base_estimator,
     n_estimators=n,
     random_state=42
  )
  model.fit(X_train, y_train)
  # Training accuracy
  train_pred = model.predict(X_train)
```

```
train_acc = accuracy_score(y_train, train_pred)
  train_scores.append(train_acc)
  # Test accuracy
  test_pred = model.predict(X_test)
  test_acc = accuracy_score(y_test, test_pred)
  test_scores.append(test_acc)
  print(f"n_estimators: {n}, Train Accuracy: {train_acc:.4f}, Test Accuracy: {test_acc:.4f}")
# Find the best number of estimators
best_n = n_estimators_range[np.argmax(test_scores)]
best_score = max(test_scores)
print(f"\nBest performance: n_estimators={best_n} with test accuracy of {best_score:.4f}")
#
    Plot
           the
                  results
plt.figure(figsize=(10, 6))
plt.plot(n_estimators_range, train_scores, label='Train Accuracy', marker='o')
plt.plot(n_estimators_range, test_scores, label='Test Accuracy', marker='o')
plt.xlabel('Number of Estimators')
plt.ylabel('Accuracy')
plt.title('AdaBoost Performance vs Number of Estimators')
plt.axvline(x=best_n, color='r', linestyle='--', label=f'Best n_estimators={best_n}')
```

```
plt.legend()
plt.grid()
plt.show()
# Train final model with best number of estimators
final_model = AdaBoostClassifier(
  estimator=base_estimator,
  n_estimators=best_n,
  random_state=42
)
final_model.fit(X_train, y_train)
# Evaluate final model
final\_pred = final\_model.predict(X\_test)
print("\nFinal Model Performance:")
print("Accuracy:", accuracy_score(y_test, final_pred))
print("\nClassification Report:")
print(classification_report(y_test, final_pred))
print("\nFeature Importances:")
feature_importances = pd.Series(final_model.feature_importances_, index=X.columns)
print(feature_importances.sort_values(ascending=False))
```

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

Program 9

			PAGE NO.
Zapas	X KA	IN .	
1210	(h1)	(211)	ald other
10/10	00	V52	- 61
1.512	Vi-25	V2125	CI
314	V13	V13	Ce
517	VET		C2
	12715	16.52	C2
4818	21.18.	MILE	Cz
35141	8.91	18.5	CZ
na.	The state of the s	+2 - (1.2)	(2:1:2)
C2 =		2-1)	0.3
	(185,14)	(39,51)	clista
(111)	0.56	3.52	CI
(1:512)	0.26	4.04	02
(314)	2.50	1-42	Cg.
(SID)	6.10	2.16	C2
(35,5)	3.82	0.41	CL
(45,5)	332	0.6)	Ca
35140)	12.13	0.21	ce
clube	(m)	1.5,0)	
	(3/4)	(2510), (351	45)
	10000		

		PAGE NO : DATE :
	Z= [-4.30535,3.7	3635,5.69305,-5.11400
	Model accuracy	
	0.0	0:8667
	Defore 0.8835	0.8667
ogiska	0.800	
-1111	- 270	0.8529
SVM	0.8728	
		0.8722
eanage	n 0.8889	1
tower		
1		
281		
मुंडाः	D+)	
	Contract of the Contract of th	
	A CONTRACTOR OF THE PARTY OF TH	

```
Code:
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
# Load your iris dataset
df = pd.read_csv("iris.csv")
# Use only Petal Length and Petal Width
X = df[["petal_length", "petal_width"]]
# Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Elbow Method to determine optimal k
inertia = []
k_range = range(1, 11)
for k in k_range:
  kmeans = KMeans(n_clusters=k, random_state=42)
  kmeans.fit(X\_scaled)
  inertia.append(kmeans.inertia_)
```

Plot Elbow Curve

```
plt.figure(figsize=(8, 5))
plt.plot(k_range, inertia, marker='o')
plt.title("Elbow Method for Optimal k")
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Inertia")
plt.grid(True)
plt.show()
# Final KMeans with optimal k (e.g., 3 from elbow)
optimal_k = 3
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
df['Cluster'] = kmeans.fit_predict(X_scaled)
# Visualize Clusters
plt.figure(figsize=(8, 5))
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=df['Cluster'], cmap='viridis', s=50)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
       color='red', marker='X', label='Centroids')
plt.xlabel("Petal Length (scaled)")
plt.ylabel("Petal Width (scaled)")
plt.title(f"K-Means Clustering (k={optimal_k}) on Iris Petal Features")
plt.legend()
plt.grid(True)
plt.show()
```

Program 10

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

	PAGE NO: DATE:
	Labo
	PCA
sol	data Matrix [4 8 13 7]
	omean center of data.
0	mean $(x_1) = \frac{9+9+13+7}{9} = 8$ mean $(x_2) = \frac{9+9+13+7}{9} = 8.5$
0	Z= e1 - Xcentue.
	X1= (0-5674) (-4) + (-0.8303) (2.5) = -4308
	Z= (0.5574)(5) + (-0.8303)(-4.5)=375
	75 = (0.5579)(5) + f0.8303)(-3.5)-5605
	14 - (05 574)(-1) + (-0.8505)(5.3) = -5.1741

		PAGE NO : DATE :
	Z= L-4-30535,3.7	3635,5.69 305, -5.11400
	Model accuracy	
	Qc Pour	After 0:8667
ogiska	Before 0.8833	0.8667
		0.8229
SVM	0.8728	
ando	n 0.8889	0.8722
tone		*
a		
28	200	
#19	20047	
-		

```
Code:
import pandas as pd
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
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.decomposition import PCA
from sklearn.metrics import accuracy_score
from sklearn.pipeline import Pipeline
# Load dataset
data = pd.read_csv('heart.csv')
# Separate features and target
X = data.drop('HeartDisease', axis=1)
y = data['HeartDisease']
# Identify categorical and numerical columns
cat_cols = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope']
num_cols = [col for col in X.columns if col not in cat_cols]
```

Label encode binary categorical columns with two categories (e.g. Sex, ExerciseAngina)

```
label_enc_cols = ['Sex', 'ExerciseAngina']
le = LabelEncoder()
for col in label_enc_cols:
  X[col] = le.fit\_transform(X[col])
# For other categorical columns with more than two categories, apply OneHotEncoding
onehot_cols = list(set(cat_cols) - set(label_enc_cols))
# Preprocessing pipeline: OneHotEncoding + scaling numerical features
preprocessor = ColumnTransformer(
  transformers=[
     ('onehot', OneHotEncoder(drop='first'), onehot_cols),
     ('scaler', StandardScaler(), num_cols)
  ],
  remainder='passthrough' # To keep label encoded columns as is
)
# Split data into train/test sets
X_train, X_test, y_train, y_test = train_test_split(
  X, y, test_size=0.3, random_state=42, stratify=y)
# Helper function to train and evaluate a model
def train_evaluate_model(model, X_train, X_test, y_train, y_test):
  pipeline = Pipeline(steps=[('preprocessor', preprocessor),
```

```
('classifier', model)])
  pipeline.fit(X_train, y_train)
  y_pred = pipeline.predict(X_test)
  acc = accuracy_score(y_test, y_pred)
  return acc, pipeline
# Train and evaluate SVM
svm = SVC(random\_state=42)
svm_acc, svm_pipeline = train_evaluate_model(svm, X_train, X_test, y_train, y_test)
# Train and evaluate Logistic Regression
logreg = LogisticRegression(max_iter=1000, random_state=42)
logreg_acc, logreg_pipeline = train_evaluate_model(logreg, X_train, X_test, y_train, y_test)
# Train and evaluate Random Forest
rf = RandomForestClassifier(random_state=42)
rf_acc, rf_pipeline = train_evaluate_model(rf, X_train, X_test, y_train, y_test)
print(f'Accuracy Scores without PCA:\nSVM: {svm_acc:.4f}\nLogistic Regression:
{logreg_acc:.4f}\nRandom Forest: {rf_acc:.4f}")
# Now apply PCA for dimensionality reduction after preprocessing (scaling + encoding)
# We modify the pipeline to include PCA before classification
def train_evaluate_model_pca(model, X_train, X_test, y_train, y_test, n_components):
```

```
pca = PCA(n components=n components, random state=42)
  pipeline = Pipeline(steps=[
     ('preprocessor', preprocessor),
     ('pca', pca),
     ('classifier', model)
  ])
  pipeline.fit(X_train, y_train)
  y_pred = pipeline.predict(X_test)
  acc = accuracy_score(y_test, y_pred)
  return acc, pipeline
# Choose number of components to keep 95% variance or fixed number (e.g. 5)
# Here let's pick 5 components arbitrarily
n_{components} = 5
svm_pca_acc, _ = train_evaluate_model_pca(svm, X_train, X_test, y_train, y_test, n_components)
logreg_pca_acc, _ = train_evaluate_model_pca(logreg, X_train, X_test, y_train, y_test,
n_components)
rf_pca_acc, _ = train_evaluate_model_pca(rf, X_train, X_test, y_train, y_test, n_components)
print(f"\nAccuracy Scores with PCA (n_components={n_components}):")
print(f"SVM: {svm_pca_acc:.4f}")
print(f"Logistic Regression: {logreg_pca_acc:.4f}")
print(f"Random Forest: {rf_pca_acc:.4f}")
```