# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.



# LAB REPORT on

# **Machine Learning (23CS6PCMAL)**

Submitted by

Abhishek S Angadi(1BM22CS007)

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 **Abhishek Sharanappa Angadi (1BM22CS007)**, 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.

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

# Index

Sl. No.	Date	Experiment Title	Page No.
1	4-3-2025	Write a python program to import and export data using Pandas library functions	5-8
2	11-3-2025	Demonstrate various data pre-processing techniques for a given dataset	9-13
3	18-3-2025	Use an appropriate data set for building the decision tree (ID3) and apply this knowledge to classify a new sample.	14-16
4	1-4-2025	Implement Linear and Multi-Linear Regression algorithm using appropriate dataset	17-20
5	8-4-2025	Build Logistic Regression Model for a given dataset	21-23
6	15-4-2025	Build KNN Classification model for a given dataset.	24-27
7	15-4-2025	Build Support vector machine model for a given dataset	28-30
8	22-4-2025	Implement Random forest ensemble method on a given dataset.	31-32
9	22-4-2025	Implement Boosting ensemble method on a given dataset.	33-35
10	29-4-2025	Build k-Means algorithm to cluster a set of data stored in a .CSV file.	
11	29-4-2025	Implement Dimensionality reduction using Principal Component Analysis (PCA) method.  40	

Github Link: https://github.com/Abbhi1234/6thSem-ML-Lab

Program 1

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

Las 1 Page No. 1	S Date:
import panda as pd	Pagé No.:
date:	importing data from specfied on the
Name Enlice Bob; Charle ]	00 00
	he path - data usv ha
"Chi - C'New York Portleton " Ch. 12	of - pd. readcs v (file-pak)
all to asso sol about this this	frint ( Marple data")
"City" - ['New York, Loudgley 'Cleage']  of pd. Dolohome (dalo)  of head ()	fle patt data (w.)  df - pd. read(sv(file-patt)  pm n/ ( staopt data")
in feed (2)	Doument dates of
A ROLL CARREST LA LA LA CO	Edos 64 St. Cotas afraganos
him him him	de policeode CSV (mobile datasel-2021. CIV)
and droped and annual and	print ("lample date")
James Age City	dl. heads) could alice
Alre 25 New York John	Comp name Model-name weger Runy
The second secon	Comp name Model-name weger RMM
import dalaset from sklown dalaset	apple iphone 16 1749 69B
Column	Front can Duction Processo Ballay Co
from stlearn daloset import loading.	12MP pl a surmp do A Thomas 16 con
from steleovo daloset import load-ins.  isis - load interes  it - pd dalofrane (ins.dato, Column- ins. feature none)	Size Carried prip ("Endo) (aun Ced )
Il - pd dalalrane Citts dato (Suran-	2024 (2024)
isse to aking more I all	Codet
m) if each is to	- Exporting Data of puller lad
106.6	
or longer = ins larget	1) dl = pd. sead : civ( cample - sales dada (s)
print ("sample dolo")	The private and the same works
dh. head ( ) .	1) Il to cru( outped (10, index valle)
The same of the same of the same of	print, " Dato saved to output cro")
Dulpul and was sounded	2) prophies
Tampe data many to the	
Canalle & Canal width mobile il policet	w. Every data the confile
100 5N 14 0.20	
Scapal length Scapal widdly polarlyll polarly	( at se of rely tass ag re by sales)
The state of the s	Lock aller to another health

	PageNo.
Analysis of Sales Dalasel patrons	Results saved to CSV lote
	Pentle Saved to Civ peter
. Sales of pd. read csv ( sales data csv)	stational phasen Cariables in the
	the date has the model good
2 tummarize cale by legin	The Real Marchese
(sale ) se 2 sale df groupby ( log ion)	National Control of the Control of t
later by & solver all many by	Ale sealth, Barelse model party
Cales by se start group by	N. E. B. D. D.
Berth 970	h c
No.14 1800	3 5
louds 1000 and alamas	PP
want was the same down	The same of the sa
louds 1000 dat algorid	The second secon
3. Grouping by product of Calculating bolas  quantity told beel selling ps - tales of.  qrouply ('Broduct')  [quantity'] Reint? Port Daliey (asserdy-  talse)	45 = [1 2 3 4] 45 = [2 4 5 4] (6 x 1 x) 1 x 1 3 4
gaenly told bed belling pr - rate -dy.	45 4 62 4597
grouphy ( Broduct )	1 12 - Part Do Kint Franch Control of the State of
Equally 7 hans 7 years Walry Caredy	Be lextro all
tale)	Stewart-out hit has a total to the
sal selling or many program	
	1 1 LY 8 - C 1 D
Product of June Visitors by the	7.0, 12
Morge 24 24	
Reploper of holy shall string	To 47 - 47 - 47 - 47 - 47 - 47 - 47 - 47
Raphopad 10	C10 32
u. Saving data to a csu file	(x,4) - 150 -10 H
	0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
Sales by-2 for CSU1' Sales by 30 (10')	E TIES
best celling to give bed selle a revit	= [ 0.1.5 -0.5] = [ 2.0 = 0.2 ]
Sales, by-re- to- CS U( 'Sales- by Re CSU') best - celling - to- CS U( 'SeV - selle - se CSU') prol("In Results loved to CSU gib")	F 5.0 (0-7
The contract of the contract o	

```
Code:
import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt
tickers = ["HDFCBANK.NS", "ICICIBANK.NS", "KOTAKBANK.NS"]
data = yf.download(tickers, start="2024-01-01", end="2024-12-30", group_by='ticker')
print("First 5 rows of the dataset:")
print(data.head())
print("\nShape of the dataset:")
print(data.shape)
print("\nColumn names:")
print(data.columns)
hdfc_data = data['HDFCBANK.NS']
print("\nSummary statistics for HDFC Bank:")
print(hdfc_data.describe())
hdfc_data['Daily Return'] = hdfc_data['Close'].pct_change()
icici_data = data['ICICIBANK.NS']
print("\nSummary statistics for ICICI Bank:")
print(icici_data.describe())
icici_data['Daily Return'] = icici_data['Close'].pct_change()
kotak_data = data['KOTAKBANK.NS']
print("\nSummary statistics for Kotak Mahindra Bank:")
print(kotak_data.describe())
```

```
kotak_data['Daily Return'] = kotak_data['Close'].pct_change()
plt.figure(figsize=(14, 10))
plt.subplot(3, 2, 1)
hdfc_data['Close'].plot(title="HDFC Bank - Closing Price")
plt.subplot(3, 2, 2)
hdfc_data['Daily Return'].plot(title="HDFC Bank - Daily Returns", color='orange')
plt.subplot(3, 2, 3)
icici_data['Close'].plot(title="ICICI Bank - Closing Price")
plt.subplot(3, 2, 4)
icici_data['Daily Return'].plot(title="ICICI Bank - Daily Returns", color='orange')
plt.subplot(3, 2, 5)
kotak_data['Close'].plot(title="Kotak Mahindra Bank - Closing Price")
plt.subplot(3, 2, 6)
kotak_data['Daily Return'].plot(title="Kotak Mahindra Bank - Daily Returns", color='orange')
plt.tight_layout()
plt.show()
hdfc_data.to_csv('hdfc_bank_data.csv')
icici_data.to_csv('icici_bank_data.csv')
kotak_data.to_csv('kotak_bank_data.csv')
print("\nHDFC Bank data saved to 'hdfc_bank_data.csv'.")
print("ICICI Bank data saved to 'icici_bank_data.csv'.")
print("Kotak Bank data saved to 'kotak_bank_data.csv'.")
```

Demonstrate various data pre-processing techniques for a given dataset

Screenshot Rob 2 Demonstrate Various data pre processing import pandas as pd de = pd. read csv (" (content / housing.c.v") print ( "Potormation of all Columns: ) print ( drinto ()) print file in statistical information of all numerical columns: 1) Orean proximity ( whomm: ") - dl. sismull () . won() columns with mining values - mining-values print (columnistivith mising, chales)

S Date	
Page Mo.	S Date :
Internation of all tolumns:	Diabetes A A A A A
Re 1 11/18	good and so some data tel
golumakon of all wolumns:	import pandar as ad
4 class : Pandas Corre for a DIR	import pandas as pd import aumpy as ap disbeles all - policiaid accordingly as ap
Danceloder: 20640 sales at	disbeles all - gel send confidenteles (sv)
Los pandas lore traine Dalobane!  Langeloder 20640 entres 0 to 20639  Data toluma (Istal to tolumy).	
	preprocessing input minmor colors
I Column Normall Court Dype	Standard of was Elong band
o longstu Ende 20640 months	import leasons as Sus
1 Calhede 20640 non-nul	import malphotals pyplot as plt
2 hours neda an 20600 nonel on	
1 Calchede 20600 nonell Gust Shype 1 Calchede 20600 nonell flate 2 housing meda as 20600 nonell my flates	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
(Modernat 10) long	de num : de select d'hypes (include - ['number'])
statistical information of all numerical id	in a less of a land of the land of
statistical inhormation of all numerical schings	imputer sample imputer ( - ctorlegy dimen)
longitutude latitude housing much age bold	The state of the s
but 20640.00000 20,640 - 20640 20640	
	7630xy de - de num quantité (0.25)
std 2.003532 2.125932 12.58558 2181.61	1352 20th - 9, -2
min -124.35 32.54 / 171 2	" Alienan
25/ -121.8 32.93 18 1447.75	11 = dl[n(cdl-numer < (0,-1,50) q e))1
50/ -118.49 24.26 29 2127	[d1-number > (0, p. 15 × 10p) 2), culoris-1
75/ -118.01 - 37.71 37 - 3148	min-mar sa ver min mor scaler ()
mox -114.31 41.95 32 39320	If minmor - pd. Dala rame (min, mor-lide.
	Ed transform (dl. nun teluno : dl-nun Colung
	Standard Scales - Standard scarter ()
	Il - Mandard - pd. Databrane ( slanded scile )
	- frankown ( dl num) Columns - dl machille
	De al Cila Descent dataset Consomer (a Mile
	print (10 broused datased (ones more (cast):

import pandas as pd import numpy as np from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import MinMaxScaler, StandardScaler from sklearn.impute import SimpleImputer from sklearn.preprocessing import LabelEncoder import seaborn as sns import matplotlib.pyplot as plt

 $\begin{aligned} & diabetes\_data = pd.read\_csv('/content/Dataset\ of\ Diabetes\ .csv')\\ & adult\_income\_data = pd.read\_csv('/content/adult.csv') \end{aligned}$ 

print("Diabetes Dataset:")
print(diabetes\_data.head())

```
print("\nAdult Income Dataset:")
print(adult income data.head())
diabetes_numerical_cols = diabetes_data.select_dtypes(include=[np.number]).columns
diabetes_categorical_cols = diabetes_data.select_dtypes(include=[object]).columns
diabetes imputer num = SimpleImputer(strategy='median')
diabetes_data[diabetes_numerical_cols] =
diabetes imputer num.fit transform(diabetes data[diabetes numerical cols])
diabetes_imputer_cat = SimpleImputer(strategy='most_frequent')
diabetes data[diabetes categorical cols] =
diabetes imputer_cat.fit_transform(diabetes_data[diabetes_categorical_cols])
adult income numerical cols = adult income data.select dtypes(include=[np.number]).columns
adult_income_categorical_cols = adult_income_data.select_dtypes(include=[object]).columns
adult_income_imputer_num = SimpleImputer(strategy='median')
adult income data[adult income numerical cols] =
adult_income_imputer_num.fit_transform(adult_income_data[adult_income_numerical_cols])
adult_income_imputer_cat = SimpleImputer(strategy='most_frequent')
adult_income_data[adult_income_categorical_cols] =
adult income imputer cat.fit transform(adult income data[adult income categorical cols])
categorical_columns_adult = adult_income_data.select_dtypes(include=['object']).columns
label encoder = LabelEncoder()
for col in categorical columns adult:
  adult_income_data[col] = label_encoder.fit_transform(adult_income_data[col])
def detect_and_remove_outliers(df):
  numerical df = df.select dtypes(include=[np.number])
  Q1 = numerical df.quantile(0.25)
  Q3 = numerical\_df.quantile(0.75)
  IQR = Q3 - Q1
  return df[\sim((numerical df < (Q1 - 1.5 * IQR)) | (numerical df > (Q3 + 1.5 * IQR))).any(axis=1)]
diabetes_data_cleaned = detect_and_remove_outliers(diabetes_data)
adult_income_data_cleaned = detect_and_remove_outliers(adult_income_data)
min max scaler = MinMaxScaler()
diabetes_numerical_cols = diabetes_data_cleaned.select_dtypes(include=[np.number]).columns
diabetes_data_normalized = diabetes_data_cleaned.copy()
diabetes data normalized[diabetes numerical cols] =
```

```
min_max_scaler.fit_transform(diabetes_data_cleaned[diabetes_numerical_cols])

adult_income_numerical_cols =
    adult_income_data_cleaned.select_dtypes(include=[np.number]).columns
    adult_income_data_normalized = adult_income_data_cleaned.copy()

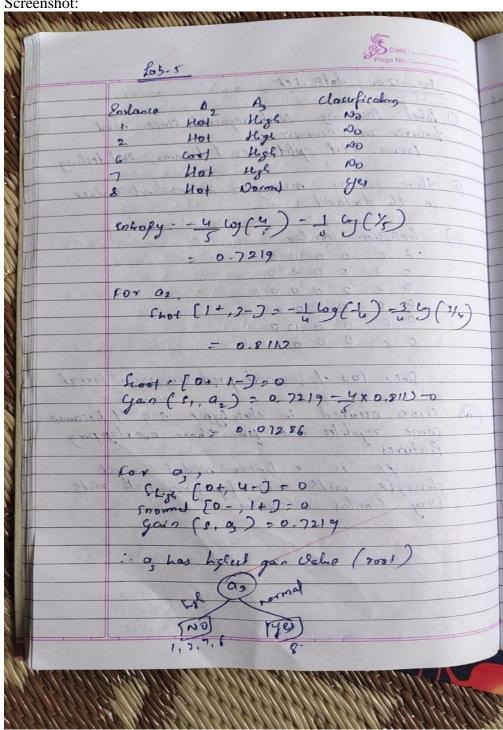
adult_income_data_normalized[adult_income_numerical_cols] =
    min_max_scaler.fit_transform(adult_income_data_cleaned[adult_income_numerical_cols])

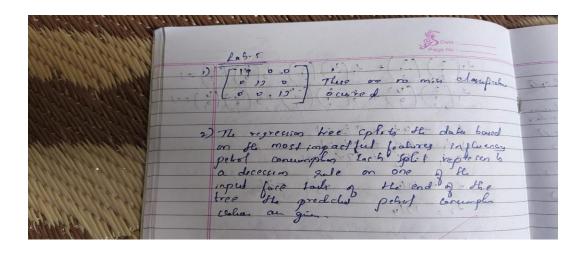
standard_scaler = StandardScaler()

diabetes_data_standardized = diabetes_data_cleaned.copy()
    diabetes_data_standardized[diabetes_numerical_cols] =
    standard_scaler.fit_transform(diabetes_data_cleaned[diabetes_numerical_cols])

adult_income_data_standardized = adult_income_data_cleaned.copy()
    adult_income_data_standardized[adult_income_numerical_cols] =
    standard_scaler.fit_transform(adult_income_data_cleaned[adult_income_numerical_cols])
```

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





```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, plot tree
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,
mean absolute error, mean squared error
from sklearn.preprocessing import LabelEncoder
iris = pd.read_csv("/content/iris (4).csv")
drug = pd.read_csv("/content/drug.csv")
petrol = pd.read_csv("/content/petrol_consumption.csv")
X_{iris} = iris.iloc[:, :-1]
y_iris = iris.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X_iris, y_iris, test_size=0.2, random_state=42)
dtc = DecisionTreeClassifier()
dtc.fit(X train, y train)
y_pred = dtc.predict(X_test)
print("Decision Tree Classification for IRIS Dataset:")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
X_drug = drug.iloc[:, :-1]
y_drug = drug.iloc[:, -1]
le = LabelEncoder()
```

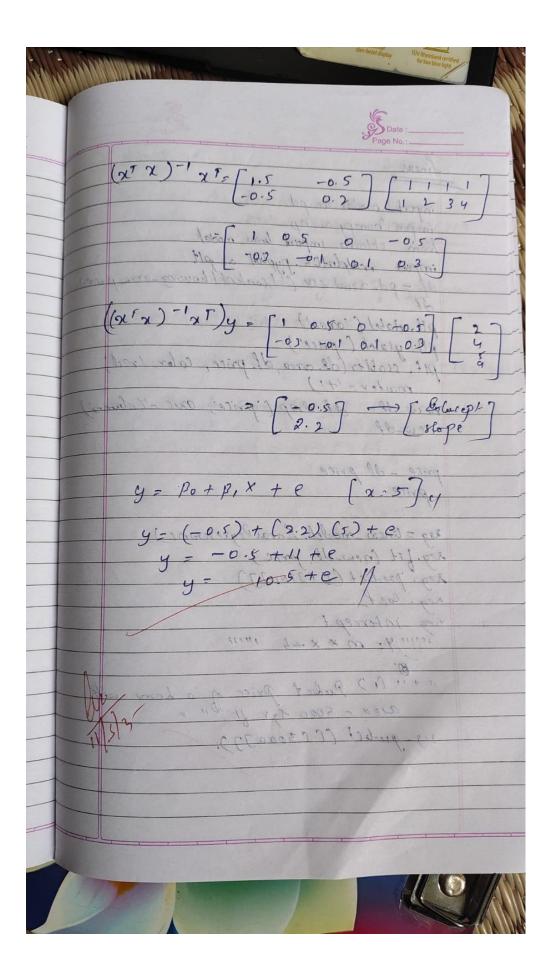
```
for col in X_drug.select_dtypes(include=['object']).columns:
  X_drug[col] = le.fit_transform(X_drug[col])
X_train, X_test, y_train, y_test = train_test_split(X_drug, y_drug, test_size=0.2, random_state=42)
dtc = DecisionTreeClassifier()
dtc.fit(X train, y train)
y_pred = dtc.predict(X_test)
print("\nDecision Tree Classification for Drug Dataset:")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
X \text{ petrol} = \text{petrol.iloc}[:, :-1]
y_petrol = petrol.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X_petrol, y_petrol, test_size=0.2, random_state=42)
dtr = DecisionTreeRegressor()
dtr.fit(X_train, y_train)
y_pred = dtr.predict(X_test)
print("\nDecision Tree Regression for Petrol Consumption:")
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error:", np.sqrt(mean_squared_error(y_test, y_pred)))
```

Program 4

Implement Linear and Multi-Linear Regression algorithm using appropriate dataset

Screenshot

11/3/25	Page No.:
	Steps: Emport stronies import date analyze data, distribution plat selationship between Variables aplet The date trans the model, predat the second, Dringbee
9	$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$
	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$



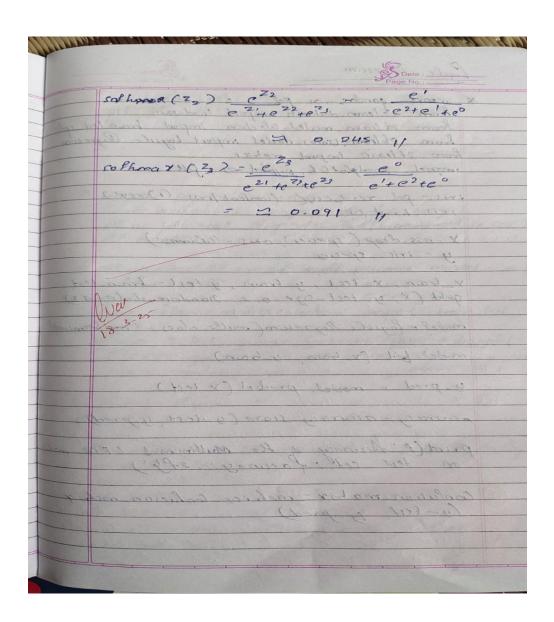
```
import pandas as pd
import numpy as np
from sklearn.model selection import train test split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean absolute error
import matplotlib.pyplot as plt
hiring_data = pd.read_csv('hiring.csv')
print(hiring_data.head())
hiring data = hiring data.dropna()
experience mapping = {
  'one': 1, 'two': 2, 'three': 3, 'four': 4, 'five': 5, 'six': 6, 'seven': 7, 'eight': 8,
  'nine': 9, 'ten': 10, 'eleven': 11, 'twelve': 12, 'thirteen': 13, 'fourteen': 14,
}
hiring data['experience'] = hiring data['experience'].replace(experience mapping)
hiring data['experience'] = pd.to numeric(hiring data['experience'], errors='coerce')
if hiring_data['experience'].isnull().any():
  print("Warning: There are still non-numeric values in the 'experience' column.")
  hiring data = hiring data.dropna(subset=['experience'])
X_hiring = hiring_data[['experience', 'test_score(out of 10)', 'interview_score(out of 10)']]
y hiring = hiring data['salary($)']
X train hiring, X test hiring, y train hiring, y test hiring = train test split(X hiring, y hiring,
test_size=0.2, random_state=42)
regressor_hiring = LinearRegression()
regressor_hiring.fit(X_train_hiring, y_train_hiring)
candidate_1 = np.array([[2, 9, 6]])
candidate_2 = np.array([[12, 10, 10]])
salary_1 = regressor_hiring.predict(candidate_1)
salary 2 = regressor hiring.predict(candidate 2)
print(f"Predicted salary for candidate 1 (2 yr experience, 9 test score, 6 interview score):
{salary_1[0]}")
print(f"Predicted salary for candidate 2 (12 yr experience, 10 test score, 10 interview score):
{salary_2[0]}")
```

```
companies data = pd.read csv('/content/1000 Companies.csv')
print(companies_data.head())
companies data = companies data.dropna()
label encoder = LabelEncoder()
companies_data['State'] = label_encoder.fit_transform(companies_data['State'])
X companies = companies data[['R&D Spend', 'Administration', 'Marketing Spend', 'State']]
y_companies = companies_data['Profit']
X_train_companies, X_test_companies, y_train_companies, y_test_companies =
train_test_split(X_companies, y_companies, test_size=0.2, random_state=42)
regressor_companies = LinearRegression()
regressor_companies.fit(X_train_companies, y_train_companies)
input_data = np.array([[91694.48, 515841.3, 11931.24, label_encoder.transform(['Florida'])[0]])
predicted profit = regressor companies.predict(input data)
print(f"Predicted profit for the given inputs (Florida State): {predicted profit[0]}")
y_pred_hiring = regressor_hiring.predict(X_test_hiring)
mae_hiring = mean_absolute_error(y_test_hiring, y_pred_hiring)
print(f"Mean Absolute Error for Salary Prediction: {mae_hiring}")
y_pred_companies = regressor_companies.predict(X_test_companies)
mae_companies = mean_absolute_error(y_test_companies, y_pred_companies)
print(f"Mean Absolute Error for Profit Prediction: {mae companies}")
```

Program 5

Build Logistic Regression Model for a given dataset

Million In	£05-3
	Logistic Regression Page No.:
0	00 5
	Producted tolong in Cardidate 1 (2 4)
	1+exp(-(-5+0.8x))
	is f(7) = 1 · dep (= (-stora 2))
	Desa Colore Penelete Cana.
1773377 F	Hereber Coor (Maje) . 1225.261
	\$(1) < 0.5 Her Ste student is
	1(1) >, 0.5 Her He studend is pas
	7, [2, 1, 0]
	softmor(Zr) = CZIE
	5-1
	solman (2): e2, = e2
	e <sup>2</sup> 1+e <sup>2</sup> 2+e <sup>3</sup> e <sup>2</sup> +e <sup>1</sup> +e <sup>0</sup>
	× 0.6652
	Carried Marie Control of the Control
1	
MINISTER !	WASHING THE WALLEST WASHING TO THE WASHINGTON OF
1141147161	



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 StandardScaler from sklearn.linear\_model import LogisticRegression from sklearn.metrics import accuracy\_score, confusion\_matrix

 $file\_path = 'HR\_comma\_sep.csv'$ 

```
data = pd.read_csv(file_path)
print(data.info())
print(data.head())
print(data.describe())
plt.figure(figsize=(8, 5))
sns.countplot(x='salary', hue='left', data=data)
plt.title('Impact of Salary on Employee Retention')
plt.xlabel('Salary')
plt.ylabel('Count')
plt.legend(title='Employee Retention', labels=['Stayed', 'Left'])
plt.show()
plt.figure(figsize=(10, 6))
sns.countplot(x='Department', hue='left', data=data)
plt.title('Impact of Department on Employee Retention')
plt.xlabel('Department')
plt.ylabel('Count')
plt.legend(title='Employee Retention', labels=['Stayed', 'Left'])
plt.xticks(rotation=45)
plt.show()
data_encoded = pd.get_dummies(data, columns=['salary', 'Department'], drop_first=True)
print(data encoded.info())
X = data encoded.drop('left', axis=1)
y = data_encoded['left']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_{\text{test\_scaled}} = \text{scaler.transform}(X_{\text{test}})
logreg = LogisticRegression(max_iter=1000)
logreg.fit(X_train_scaled, y_train)
y_pred = logreg.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of the Logistic Regression Model: {accuracy * 100:.2f}%")
cm = confusion matrix(y test, y pred)
```

```
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False, xticklabels=['Stayed', 'Left'],
yticklabels=['Stayed', 'Left'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

Build KNN Classification model for a given dataset.

	Las-6 KNN Page No.:
1	Person so Age of Salony Tongeto
V	10 PM 23 77 51 85 A1 8
5	1618 9 24 06 90 PE N 3
18	De 141 00 60 14 4 6
1	MEIC 440 00 90 ENY 3
1	E 09 436 ando 85 À 1
	× 135 00 100 682 V
	Par (19,50) d = (19,50)
	For (19, 50) d = \(\text{\$\tau_2-\text{\$\text{\$\general}{\general}
	(B) F(C)
	V 7:= (52.27) x 02
	For (23,50); $\sqrt{(35-23)^2+(100-50)^2}$ = 46.57
	= 46.57
	Coc 12 12 12 12 12 12 12 12 12 12 12 12 12
	for (34, 70): 5 (35-41)2+ (100-60)2
	2 40.44
	for (4), 70): \( 36-43)^2 + (100-30)2
	= 31.04
	Por (39, 40)= V(35-30)2+(100-40)2
	= 60.07.

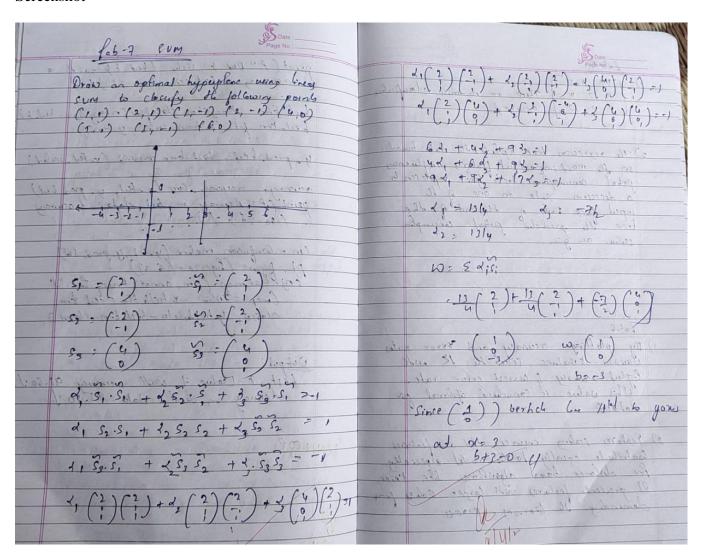
				Page		
	D Y	4) 00 38 000	60 1N 20 (S)	y	31.95 40.44 31.04	Rat 5 4 2 3 1 6
	s (re	2 2-9 1 - 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	TM & C	11 50	1 m2	
		35-22)24 35-41)24 40-612		2 5	100 (a)	
2 (		1(35-38) = +			for (1)	
	A	C6 03				

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy score, confusion matrix, classification report
import matplotlib.pyplot as plt
import seaborn as sns
iris_df = pd.read_csv('/content/iris (3).csv')
print(iris_df.head())
X iris = iris df.drop(columns=['species'])
y_iris = iris_df['species']
X_train_iris, X_test_iris, y_train_iris, y_test_iris = train_test_split(X_iris, y_iris, test_size=0.2,
random state=42)
scaler = StandardScaler()
X_train_iris = scaler.fit_transform(X_train_iris)
X_test_iris = scaler.transform(X_test_iris)
knn_iris = KNeighborsClassifier(n_neighbors=3)
knn iris.fit(X train iris, y train iris)
y pred iris = knn iris.predict(X test iris)
accuracy_iris = accuracy_score(y_test_iris, y_pred_iris)
print(f"Accuracy on Iris test data: {accuracy_iris * 100:.2f}%")
cm_iris = confusion_matrix(y_test_iris, y_pred_iris)
sns.heatmap(cm_iris, annot=True, fmt="d", cmap="Blues", xticklabels=knn_iris.classes_,
yticklabels=knn_iris.classes_)
plt.title("Confusion Matrix for Iris Dataset")
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
print("Classification Report for Iris Dataset:")
print(classification_report(y_test_iris, y_pred_iris))
diabetes_df = pd.read_csv('diabetes.csv')
print(diabetes_df.head())
```

```
X_diabetes = diabetes_df.drop(columns=['Outcome'])
y_diabetes = diabetes_df['Outcome']
X_train_diabetes, X_test_diabetes, y_train_diabetes, y_test_diabetes = train_test_split(X_diabetes,
y_diabetes, test_size=0.2, random_state=42)
scaler = StandardScaler()
X train diabetes = scaler.fit transform(X train diabetes)
X_{\text{test\_diabetes}} = \text{scaler.transform}(X_{\text{test\_diabetes}})
knn_diabetes = KNeighborsClassifier(n_neighbors=5)
knn_diabetes.fit(X_train_diabetes, y_train_diabetes)
y_pred_diabetes = knn_diabetes.predict(X_test_diabetes)
accuracy_diabetes = accuracy_score(y_test_diabetes, y_pred_diabetes)
print(f"Accuracy on Diabetes test data: {accuracy diabetes * 100:.2f}%")
cm diabetes = confusion matrix(y test diabetes, y pred diabetes)
sns.heatmap(cm_diabetes, annot=True, fmt="d", cmap="Blues", xticklabels=knn_diabetes.classes_,
yticklabels=knn_diabetes.classes_)
plt.title("Confusion Matrix for Diabetes Dataset")
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
print("Classification Report for Diabetes Dataset:")
print(classification_report(y_test_diabetes, y_pred_diabetes))
```

Build Support vector machine model for a given dataset

#### Screenshot

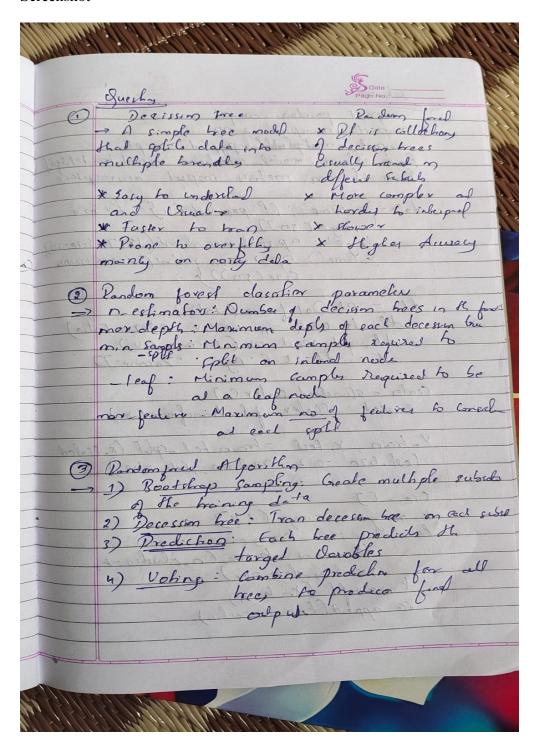


# Code:

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, roc\_auc\_score, roc\_curve
from sklearn.preprocessing import LabelEncoder, label\_binarize
import matplotlib.pyplot as plt
import numpy as np

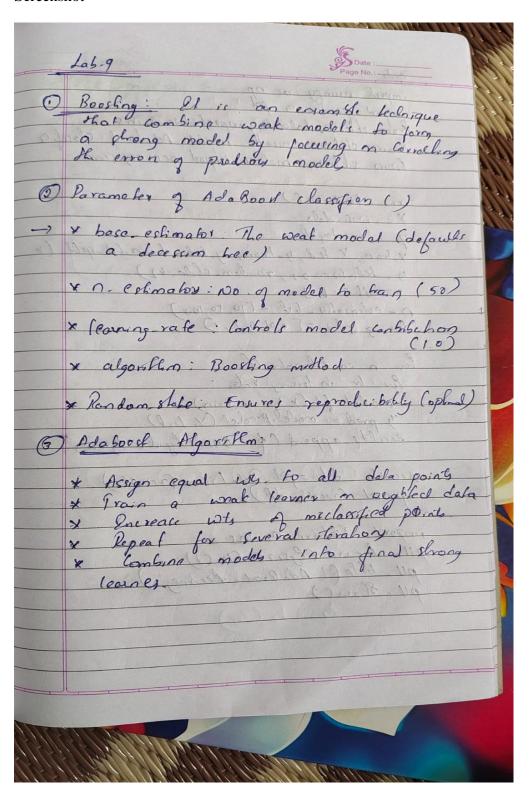
```
import seaborn as sns
df = pd.read csv("/content/letter-recognition.csv")
top_classes = df['letter'].value_counts().head(5).index.tolist()
df = df[df['letter'].isin(top_classes)]
X = df.iloc[:, 1:]
y = df.iloc[:, 0]
label encoder = LabelEncoder()
y encoded = label encoder.fit transform(y)
y_bin = label_binarize(y_encoded, classes=np.unique(y_encoded))
n_{classes} = y_{bin.shape[1]}
X_train, X_test, y_train, y_test_bin = train_test_split(X, y_bin, test_size=0.2, random_state=42)
svm_model = SVC(kernel='linear', probability=True)
svm model.fit(X train, y train.argmax(axis=1))
y_score = svm_model.predict_proba(X_test)
y_pred = svm_model.predict(X_test)
y_true = y_test_bin.argmax(axis=1)
print("Accuracy:", accuracy_score(y_true, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_true, y_pred))
plt.figure()
for i in range(n classes):
  fpr, tpr, _ = roc_curve(y_test_bin[:, i], y_score[:, i])
  auc = roc_auc_score(y_test_bin[:, i], y_score[:, i])
  plt.plot(fpr, tpr, label=f"{label_encoder.inverse_transform([i])[0]} AUC={auc:.2f}")
plt.plot([0, 1], [0, 1], 'k--')
plt.title("ROC Curve (Top 5 Classes)")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend(loc="lower right")
plt.tight_layout()
plt.show()
macro_auc = roc_auc_score(y_test_bin, y_score, average="macro")
print("Macro AUC Score:", macro_auc)
```

Implement Random forest ensemble method on a given dataset.



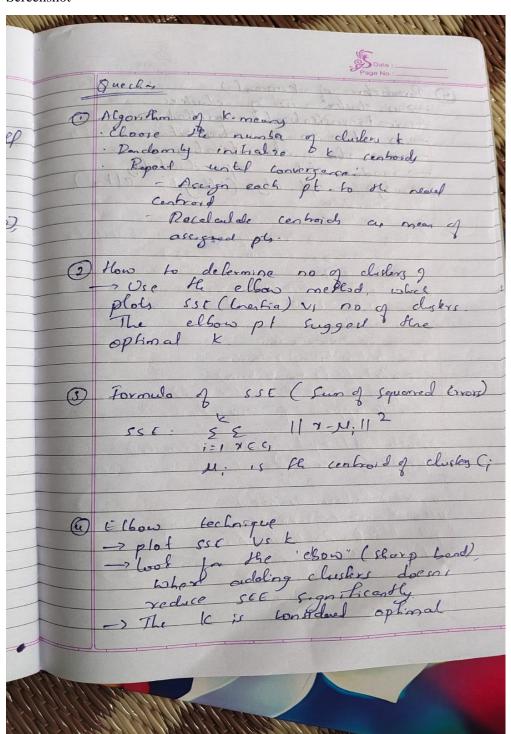
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn import preprocessing
df = pd.read_csv('/content/train.csv')
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
for column in X.columns:
  if X[column].dtype == 'object':
   le = preprocessing.LabelEncoder()
   X[column] = le.fit\_transform(X[column])
if y.dtype == 'object':
 le = preprocessing.LabelEncoder()
 y = le.fit_transform(y)
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)
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
print(f"Accuracy: {accuracy}")
print(f"Confusion Matrix:\n{conf_matrix}")
```

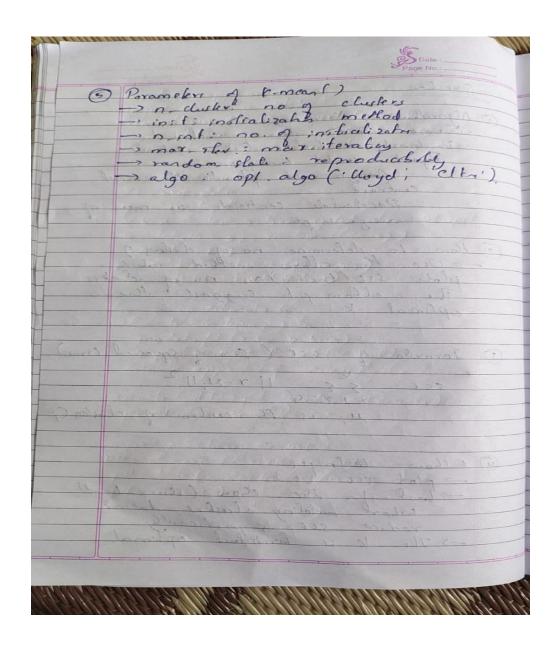
Implement Boosting ensemble method on a given dataset.



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load iris
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy score
iris = load iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
results = []
n_{estimators_list} = [10, 50, 100]
learning_rates = [0.01, 0.1, 1]
for n in n_estimators_list:
  for lr in learning_rates:
     tree_base = DecisionTreeClassifier(max_depth=1)
     model = AdaBoostClassifier(estimator=tree_base, n_estimators=n, learning_rate=lr,
random_state=42)
     model.fit(X_train, y_train)
     y_pred = model.predict(X_test)
     acc = accuracy_score(y_test, y_pred)
     results.append({
       'Base': 'DecisionTree',
       'n_estimators': n,
       'learning_rate': lr,
       'Accuracy': acc
     })
for n in n_estimators_list:
  for lr in learning_rates:
     log_reg_base = LogisticRegression(max_iter=1000)
     model = AdaBoostClassifier(estimator=log_reg_base, n_estimators=n, learning_rate=lr,
random state=42)
     model.fit(X_train, y_train)
     y_pred = model.predict(X_test)
     acc = accuracy_score(y_test, y_pred)
```

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

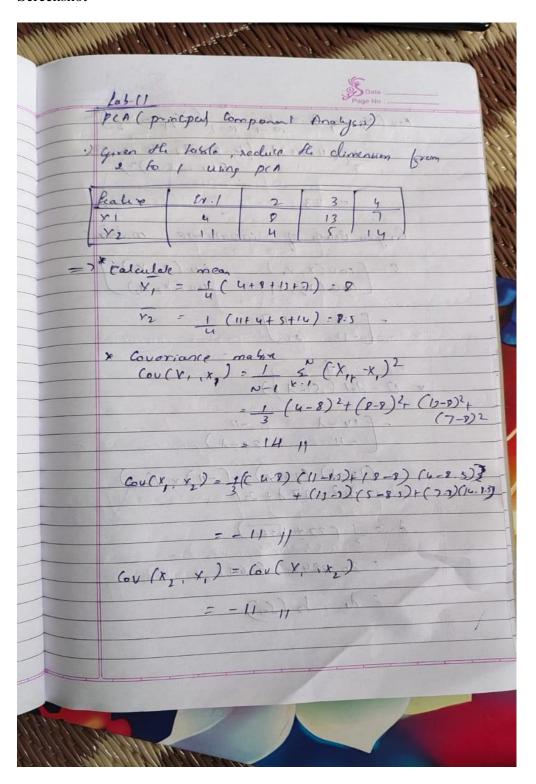




import pandas as pd import numpy as np import matplotlib.pyplot as plt from sklearn.cluster import KMeans from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import StandardScaler from sklearn.metrics import accuracy\_score

```
data = {
  'Name': [f'Person_{i+1}]' for i in range(50)],
  'Age': np.random.randint(18, 70, size=50),
  'Income': np.random.randint(20000, 120000, size=50)
}
df = pd.DataFrame(data)
df.to_csv('income.csv', index=False)
df = pd.read_csv('income.csv')
X = df[['Age', 'Income']]
scaler = StandardScaler()
X scaled = scaler.fit transform(X)
X_train, X_test = train_test_split(X_scaled, test_size=0.2, random_state=42)
sse = []
k_range = range(1, 11)
for k in k_range:
  kmeans = KMeans(n_clusters=k, random_state=42)
  kmeans.fit(X_train)
  sse.append(kmeans.inertia_)
plt.plot(k_range, sse, marker='o')
plt.title('SSE vs Number of Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('Sum of Squared Errors (SSE)')
plt.show()
optimal_k = 3
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
kmeans.fit(X_train)
y_pred = kmeans.predict(X_test)
print(f'Predicted Clusters for Test Data: {y_pred}')
```

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



$\begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} $	(14-1) (1-1)
= h, (b9)	11 12, 11 = V 11 = +(16-24,)2 = 12+(10-30, 700)2 = 19.7348

a voit eigenvecht con to die e, = [1/1/4/11 (14-1,) / 11 4,1) (14-30.3849) 1(19.7548) 7 0.5574 \* Compulation of sel principle component be the KK sample in the above fable (datel) I'm principle componed of the or is given by e, T f x1r - x, ] [0.5574-0.830] [144] - 0.5574(8,,-x,)-

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.decomposition import PCA
from scipy import stats
df = pd.read_csv('heart (2).csv')
z_scores = np.abs(stats.zscore(df.select_dtypes(include=[np.number])))
df_{no}outliers = df[(z_{scores} < 3).all(axis=1)]
df_cleaned = df_no_outliers.copy()
for col in df cleaned.select dtypes(include='object').columns:
  df_cleaned[col] = LabelEncoder().fit_transform(df_cleaned[col])
X = df cleaned.drop('HeartDisease', axis=1)
y = df_cleaned['HeartDisease']
scaler = StandardScaler()
X_{scaled} = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42,
stratify=y)
models = {
  "Logistic Regression": Logistic Regression (max iter=1000),
  "Random Forest": RandomForestClassifier(),
  "SVM": SVC()
}
print("Accuracy without PCA:")
for name, model in models.items():
  model.fit(X_train, y_train)
  y pred = model.predict(X test)
  acc = accuracy_score(y_test, y_pred)
  print(f"{name}: {acc:.4f}")
pca = PCA(n\_components=5)
X_pca = pca.fit_transform(X_scaled)
X_train_pca, X_test_pca, y_train, y_test = train_test_split(X_pca, y, test_size=0.2, random_state=42,
```

```
stratify=y)

print("\nAccuracy with PCA:")
for name, model in models.items():
    model.fit(X_train_pca, y_train)
    y_pred = model.predict(X_test_pca)
    acc = accuracy_score(y_test, y_pred)
    print(f"{name}: {acc:.4f}")
```