VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.



Machine Learning (23CS6PCMAL)

Submitted by

Sneha N Shastri (1BM22CS283)

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 **Sneha N Shastri (1BM22CS283)** 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.

Lab Faculty Incharge

Name: Ms. Sunayana S 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
2	3-3-2025	Demonstrate various data pre-processing techniques for a given dataset	
3	10-3-2023	Implement Linear and Multi-Linear Regression algorithm using appropriate dataset	
4	17-3-2025	Build Logistic Regression Model for a given dataset	12
5	/ 4-7-/1//	Use an appropriate data set for building the decision tree (ID3) and apply this knowledge to classify a new sample	
6	7-4-2025	Build KNN Classification model for a given dataset	24
7	21-4-2025	Build Support vector machine model for a given dataset	28
8	5-5-2025	Implement Random forest ensemble method on a given dataset	31
9	5-5-2025	Implement Boosting ensemble method on a given dataset	36
10	12-5-2025	Build k-Means algorithm to cluster a set of data stored in a .CSV file	39
11	1/- >-/(1/-)	Implement Dimensionality reduction using Principal Component Analysis (PCA) method	43

Github Link:

https://github.com/snehanshastri/ML

Program 1

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

```
55 03 2025
Quiro 11 mothods to read data
infor numpy as up
# 1 - directly add data
  $ 1USN' : [ 280, 281, 287, 283, 284]
   'Name! [ 'allen', 'Authory', 'Akbar, 'Amas',
   1 marks 1: [80,90,95,91.93]
H=pd. Dataframe (dass)
29
    Alen 80
280
     anthony 90
#2 - From sklearn datasets
from skleam datasets import load stabetes
diabeter = load - diabeters
of = pcl . Data Frame (dissets data, columns = diabetes.
                teature -names)
MI 'target' ] = diabeter . target
Print ("Sample data:")
print ( of head ( ))
```

```
: #8 - Reading from CCV
                                                 hate - data [ close 1] . plot (+the = " HOTE - closing
  1/2 = pot-read - csv ( surple - sales - data . cevil
                                                  pt. show()
  1/2
                                                 of similarity for kotak and Icici
 Sample Butput
                         Price Sales
      Product Quentity
                         1000 5000
                                                  botte - data ['Daily - Return 1]. plot (title = 1
                                        NOLK
                5
    laptop
                                       West
                15
                                 300
                                                  pir. shows)
    Mouse
                                                 I similarly for kotak and Icici
#4 - From kaggle
 olis: pd. read - (sv ( Diabeter - (sv))
                                                  Durput - Line Plots for Clasing Price and Daily
 of 3- head ()
Using Yahoo Finance API
 import ytirance as yf
 import marphothis pyplor as plt
 Hickey - [ " H DFCBANK . NS" 4 ICICIBANK . NS"
                      " KOTAKBANK . NS" ]
dara = yf. download (tickers, start = 12024-01-01
                         end=112024-12-804
                         group - by : ticker )
print (data. head ())
 Lage - data = data [ HOFMANK. NS']
 kotak data = data[ KOTA KBANK . NS/]
 icici-data = data [ 1 ICICI BANK . NS!]
```

```
import pandas as pd
import numpy as np
data={
```

```
'USN': [280,281,282,283,284],
    'Name':['Alex','Anthony','Akbar','Amar','Asha'],
    'Marks': [80,90,95,91,93]
df=pd.DataFrame(data)
df
from sklearn.datasets import load diabetes
diabetes = load diabetes()
df = pd.DataFrame(diabetes.data,
columns=diabetes.feature names)
df['target'] = diabetes.target
print("Sample data:")
print(df.head())
df2=pd.read csv('/content/sample sales data.csv')
df2
df3=pd.read csv('/content/Dataset of Diabetes .csv')
df3.head()
import yfinance as yf
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())
hdfc data=data['HDFCBANK.NS']
kotak data=data['KOTAKBANK.NS']
icici data=data['ICICIBANK.NS']
hdfc data['Close'].plot(title="HDFC- Closing Price")
plt.show()
kotak data['Close'].plot(title="Kotak Bank - Closing Price")
plt.show()
icici data['Close'].plot(title="ICICI Bank - Closing Price")
plt.show()
hdfc data['Daily Return'] = hdfc data['Close'].pct change()
kotak data['Daily Return'] = kotak data['Close'].pct change()
icici data['Daily Return'] = icici data['Close'].pct change()
```

```
hdfc_data['Daily Return'].plot(title="HDFC - Daily Return")
plt.show()
kotak_data['Daily Return'].plot(title="Kotak Bank - Daily
Return")
plt.show()
icici_data['Daily Return'].plot(title="ICICI Bank - Daily
Return")
plt.show()
```

Demonstrate various data pre-processing techniques for a given dataset

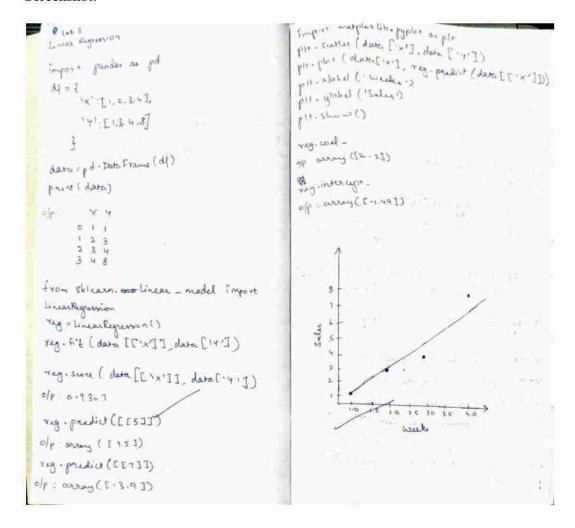
```
was ing colo-conduct = offer entirences [ offer is much to
05/03/2025
                                                  is sing-cels-diabetes-dfs-columns [dfs. issuer)
 Lob 1 .
 import poundar or pol
 Import numby as of
 dispoling car car (1 housing car)
                                                 y Sinding Categorical columns
                                                  pregonal-colo-adese = decesalect - dogres
(include = Probject of column
# display into of all columns
# display statistical info
                                                 aregorical-colo-de alceles: df5. selece -drypes
(include = ( object 13) . column
  dl. describe()
# display cow of unique labels for ocean
 promining column
 print ( oil [ ocean - proximity ] . Value - county)
                                                 O. Difference blow mu Man Sealing and
  print of [ acan proximity ] - num que ()
                                                    Standardization
                                                         Min Mark Scaling Standardization
# Cours with missing values >0
                                                    _ Rescales date to a - Transfer me data
 missing -cots off-columns [ dl. is need (). sum (), o
                                                    fixed range wouldy to have mean of a
 missing - col
                                                    [1,1-] ro [1,0]
                                                                            and so of 1
# Pre- processing Diabeter and Adult Income
                                                   -> Use last when -> when date follows
 df5=pd. read-cov ("Datnew of trabety". (sv)
                                                    we need to maintain thermas distribution
                                                     original distribution
                                                                             or has outliers
                                                    and have known
 dib:pd. read - csv ( adult. csv )
# Finding mi columns with missing values
```

```
Code:
```

```
missing_col=df.columns[df.isnull().sum()>0]
missing_col
# For 'adult.csv' (df4)
```

```
missing cols adult = df4.columns[df4.isnull().sum() > 0]
print(f"Columns with missing values in 'adult.csv':
{missing cols adult.tolist()}")
   # For 'Dataset of Diabetes .csv' (df5)
missing cols diabetes = df5.columns[df5.isnull().sum() > 0]
print(f"Columns with missing values in 'Dataset of Diabetes
.csv': {missing cols diabetes.tolist()}")
# For 'adult.csv' (df4)
categorical cols adult =
df4.select dtypes(include=['object']).columns
print(f"Categorical columns in 'adult.csv':
{categorical cols adult.tolist()}")
   # For 'Dataset of Diabetes .csv' (df5)
categorical cols diabetes =
df5.select dtypes(include=['object']).columns
print(f"Categorical columns in 'Dataset of Diabetes .csv':
{categorical cols diabetes.tolist()}")
```

Implement Linear and Multi-Linear Regression algorithm using appropriate dataset



```
Linear Eigenvion - Marie
                                               [2.23]
   Tompose pundos as pot
                                              y = betalolo] + bun [] + ex
   impose sumpy as of
   clare = E
                                              y
      1x1.11.2,5,4]
       [3,4,2,1].14
                                                wray ( [[0.7,0.7]
                                                         [0.7,2.9]
   de = pd. DataFrame (data)
                                                         [0.7,5.1]
    print (df)
   x= np. array (4['x']). Yeshaper (-Li)
                                                         ([[E. r. p. 0]
   X = np. concar enate ((x, np. ones ((x. shape [o]
                                             impost matriotist. pyriot on pin
                                              pit. scarles (de L'x 17, de [141])
                              1) ), anci = 1
                                              pir. plo+ (xt: 13, yt: 13, 17)
  X = hp. concatenate((hp.onas((v.shapePo], 3)
  Y= 40. array (d1[141]). restage (-1,1)
  X - transpose = np. Hamepose (x)
  XTX = hy. dot (x transpose ,x)
  XTX-inv=np. linalg. inv (XTX) ...
  XTX-Inv-X-transpore = mp. dar (xTX-inv,
 beta: np. dot (xTx-inv-x-hangeose, y)
 print (beton)
```

```
import pandas as pd
import numpy as np
data={
   'X':[1,2,3,4],
```

```
'Y':[1,3,4,8]
df=pd.DataFrame(data)
print(df)
X=np.array(df['X']).reshape(-1,1)
X=np.concatenate((X,np.ones((X.shape[0],1))),axis=1)
X = np.concatenate((np.ones((X.shape[0], 1)),
X[:,0].reshape(-1,1)), axis=1)
Y=np.array(df['Y']).reshape(-1,1)
print(X)
print(Y)
X_transpose = np.transpose(X)
XTX = np.dot(X transpose, X)
XTX inv = np.linalg.inv(XTX)
XTX inv X transpose = np.dot(XTX inv, X transpose)
beta = np.dot(XTX inv X transpose, Y)
print(beta)
```

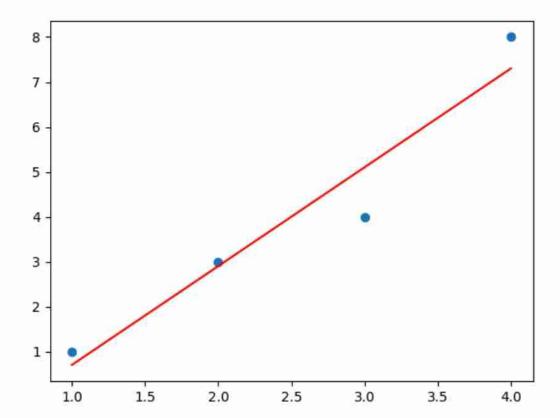
```
y=beta[0]+beta[1]*X

Y

import matplotlib.pyplot as plt

plt.scatter(df['X'],df['Y'])

plt.plot(X[:,1],y[:,1],'r')
```



Program 4 Build Logistic Regression Model for a given dataset

Legistic Registeries wany tric December	user
Electronia assert	world sit (X - man, y - train)
Logistic	The same of the same of the same of
and punker int pt	proble - State (K - Mat, y - Ena)
Empere punder as pl	e/f 6-9313
From stellars darmen Tomport local 125	LA fre
From skleam-darmen import that	male - (202)
5 F 405 4	(Fall Derive
from oblesen support metrics	annak Otto in mark to
from sklear inform	belown
Inix = lend -ivix()	
as pH	con : confusion - matrice (y - text)
import matphentils-pyphot as pt	arme (4-tet+
	em : confusion - matrix (y - text , y - producted)
divito(s)	o/c array (III, 0, 0]
of Coesce.	Terority (
4	[0, 11, 13]
latel,	[o, 1, 1017]
desa-module,	
(a constant	ingost beaton as on
1 femmes - names,	pit. Engrave (Program
to to	sur heatmap (em connot = True)
fileran 10	of x label (Predicted)
maget,	
	pre- glabel ('Truther)
touget_rames]	
	Binary Clarification - HR dataset
Lolardo ziri	The paraget
ole: way (E5.1,3.5, 1.4, 0.21)	Or Concider.
	2 5 - 21 - 0 - 8 Z = 0 + 2, x
from schem . linear model import Logistic	Z = 0. + 0.x
Reguesion	
and the second second	= -5 + 0.8%
model = Logistic Registrion ()	a. Function = $\frac{1}{1+e^{-2}} = \frac{1}{1+e^{-(-5+o-1x)}}$
X-train, X-test, y-train, y-test = train-test.	1+e-2 1+p-(-8+0-8x)
X-train, X-text, y-train, y-test = train-text. equit (tris. data, it is. tenger, text - size = 0.2)	

```
" Did her brilan and good briles comed sech;
                 = 1 = 5-66 = 1000.6
                                                                                                                                                       enecking for trul values : de consult : Sumes
                                                                                                                                                      Gues Dropping animal- harm column
                 = 0-6 = 0.5 | hence Pours
                                                                                                                                                        pr=d1-drop( animal - name ', com = 1)
                                                                                                                                                o were there my nice ing values
     2. 2 = [2,1,0]
                                                                                                                                                  3. What does the confusion meeters tell
                                                                                                                                                       about the model performances
                                                                                                                                                           It shows the true positive, the negative,
                                                                                                                                                         Solu positive, talse negative
       Softman (21) = 21 (216,400 = 0.244
                                                                                                                                                 is which class types were most frequently
                                                                                                                                                     musclassofied?
      Coltumn (20) = 60 = 200001
Dwich variables have a direct and clean
                                                                                                                                                                                     all the state of t
       impact on employee retaining why?
                                                                                                                                                                                            A Starte In the
       -> Satisfaction-lefted : high satisfaction, to w leaving
        - sureage-monthly-hours: higher no of hour, high
       - promotion-lest - syches: low promotion, h
                                                                                                                                                                                                       and the second of the second of
        - salary : lower scalary , higher leaving rate
 (2) what Is the occurring of the model!
                                                                                                                                                                     The state of the state of the state of
                                                                                                                                                                     to and a lighting of a right
```

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

```
import matplotlib.pyplot as plt
df=pd.read csv('/content/zoo-data (3).csv')
df.head()
df.isnull().sum()
df=df.drop('animal name',axis=1)
df.head()
target=df['class type']
data=df.drop('class type',axis=1)
data.head()
target.head()
from sklearn.model selection import train test split
x train,x test,y train,y test=train test split(data,target,tes
t size=0.2)
from sklearn.linear model import LogisticRegression
model=LogisticRegression()
model.fit(x train,y train)
from sklearn import metrics
model.score(x test,y test)
model.predict(x test)
from sklearn.metrics import confusion matrix
```

```
# Assuming 'y_test' and 'predictions' are your true labels and
model predictions

predictions = model.predict(x_test)

cm = confusion_matrix(y_test, predictions)

print(cm)

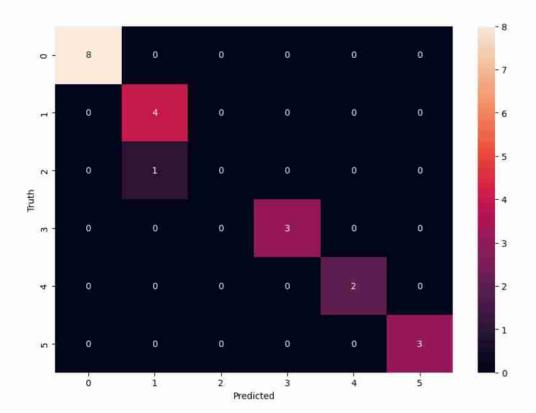
import seaborn as sn

plt.figure(figsize = (10,7))

sn.heatmap(cm, annot=True)

plt.xlabel('Predicted')

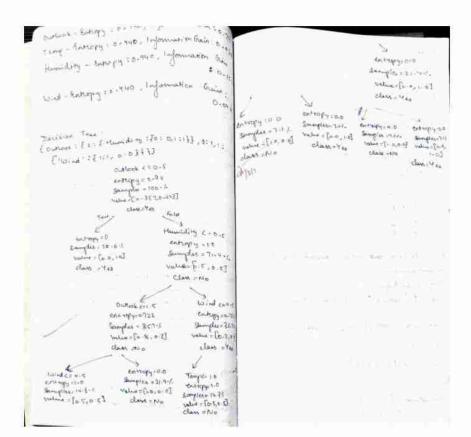
plt.ylabel('Truth')
```



Program 5
Use an appropriate data set for building the decision tree (ID3) and apply this knowledge to classify a new sample

	ANDRES - SYLVEN
12/03/2025 edges ithin for weather dataset	If biological (=0 spr 0 poly (histories)
Lab 2 = 103 sulger 11	Letter Sent Topy
import pandas as pol from sklean tree import Decision Tree Claudes,	dul calculate - Information - quin (data, transe, targer - column):
from skillar proper as pit	targer - column):
Import matrices	to trul - entropy = calculate - entropy (class o
Empore mate preprocessing Emport Label Encoder	total-entropy = calculate-entropy (place, o targer-column)
from	(carry con 1)
df=pd-xend-cev("nocather-data") df, head()	meighted -ent ropy = 0
front de = de drop ([10 ay :], anis = 1)	for value in whigh - value: Subset = data [data[feature] = - value]
final-of. head ()	proportion = 1 en (culvet)/len (data)
y= 44 + 0 x cisio 4	weighted - entropy + = proportion +
encoder. (abrith coder ()	- enropy (subser, touget_ column
for col in final-di-columns;	information-gain = total-entropy-weighted
Fried-dy [col3 - encoder. fit - transferm (d) [col]	- entropy
def culculate en vopy (data, torger_column):	raturn information-gain
total rous elin (data)	del id3 (data, target-column, Pentures):
tunger values = data [tunget = column]. unique ()	I len(data [turget -column].unique())==1:
for value in target - values:	Vitues data[taget-column].meds ().
warms - wound = ten (data [data [data [+ engel - co and	
proportion trace - count / to tul-rous	bast-feature = max (features, key = landa 2. controllate_information_gain (date, 2, target_common)

```
171.2x = 1 6 xx +-- -
                                                               hous, busin')
        remaining-frameworth for f in freq
                                                     parts (1" { 1 samuely - housepy : { energy : 10}
                      A F ! = but Penting ]
                                                           Information Come [ infogram . - Et ] )
        for value in dura [ bed - feature] . uniq
                                                   Lines with list (final - df - columns , -13)
                                                   langue - Education = " Decision !
          subset a data (data) best - feature ] =
                                                   desirion tree - 1013 (comes of , torque when , frances)
                                                   point (" In Decision Taxe ")
          Jam (Subset ) = = 0:
                                                   point (places on - tran)
               Pres [but - Factor I [vome] = dage
                                                   ell - be a wont were Chamily as ( outtoner elentrapy)
                            [ terry et - column].
                               much (). itactos
                                                   X = first - OH[frankley]
                                                   y - final -df[tunget - column]
         والعالو
                                                   d1. 4. +(x-4)
            Epi - [ but I same I water ] - id 3
                                                   pu. Eguns ( figsise = (14, 103)
                (subset, truget - tolumn,
                                remaining francy
                                                   plor-nee (the feature name of eatures.
                                                               class names = [ " No", "Yes "].
       return their
                                                               filled strue, rounded - True
energy : culculate - energy ( final of , Drigging)
                                                               fonticise = 10, proportion = True )
bery ( f. , " Emile & Breise : formold: -3+7 /4)
                                                   pla. eles es ()
print ( Feature wise Survey and leformation
                                                   Output:
        ( : " )
                                                    Emply of Decision: 0.940
for France in Final-Africhamous [:-1] -
     entropy = counted - entropy (final-cht, Decision)
                                                    Feature wise extrapy and information
     into gain - cultural - information - gain ( final )
                                                   and de Guin :
```



```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, plot tree
import matplotlib.pyplot as plt
import math
from sklearn.preprocessing import LabelEncoder
df=pd.read csv('/content/weatherdata.csv')
df.head()
final df=df.drop(['Day'],axis=1)
final df.head()
y=df.Decision
# Encode categorical features into numeric values for
DecisionTreeClassifier
encoder = LabelEncoder()
for col in final df.columns:
    final df[col] = encoder.fit transform(df[col])
```

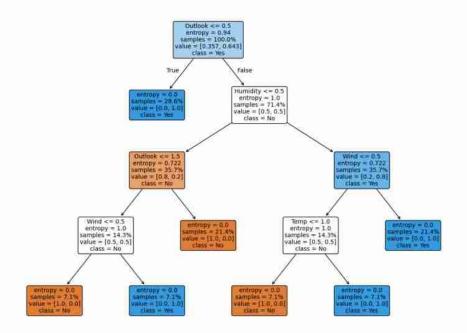
```
def calculate entropy(data, target column):
    total rows = len(data)
    target values = data[target column].unique()
    entropy = 0
    for value in target values:
       value count = len(data[data[target column] == value])
       proportion = value count / total rows
        entropy -= proportion * math.log2(proportion) if
proportion != 0 else 0
    return entropy
def calculate information gain(data, feature, target column):
    total entropy = calculate entropy(data, target column)
    unique values = data[feature].unique()
    weighted entropy = 0
```

```
for value in unique values:
        subset = data[data[feature] == value]
       proportion = len(subset) / len(data)
       weighted entropy += proportion *
calculate entropy(subset, target column)
    information gain = total entropy - weighted entropy
    return information gain
# Function to implement the ID3 algorithm
def id3(data, target column, features):
    # If all target values are the same, return that value
    if len(data[target column].unique()) == 1:
       return data[target column].iloc[0]
    # If no features left, return the most common target value
    if len(features) == 0:
        return data[target column].mode().iloc[0]
    best feature = max(features, key=lambda x:
```

```
calculate information gain(data, x, target column))
    # Create a new tree node
    tree = {best feature: {}}
    # Remove the best feature from the list of available
features
    remaining features = [f for f in features if f !=
best feature]
    for value in data[best feature].unique():
        subset = data[data[best feature] == value]
       if len(subset) == 0:
            tree[best feature][value] =
data[target column].mode().iloc[0]
       else:
            tree[best feature][value] = id3(subset,
target column, remaining features)
    return tree
 Display entropy of target column
```

```
entropy = calculate entropy(final df, 'Decision')
print(f"\nEntropy of Decision: {entropy:.3f}\n")
# Display information gain for each feature
print("Feature-wise Entropy and Information Gain:")
for feature in final df.columns[:-1]:
    entropy = calculate entropy(final df, 'Decision')
    info gain = calculate information gain(final df,
feature, 'Decision')
    print(f"{feature} - Entropy: {entropy:.3f}, Information
Gain: {info gain:.3f}")
# Build decision tree using ID3
features = list(final df.columns[:-1]) # All columns except
the target
target column = 'Decision'
decision tree = id3(final df, target column, features)
print("\nDecision Tree:")
print(decision tree)
 --- Train and plot using DecisionTreeClassifier for
```

```
visualization ---
clf = DecisionTreeClassifier(criterion='entropy')
# Prepare data for the classifier
X = final df[features]
y = final df[target column]
clf.fit(X, y)
# Plot the complete decision tree
plt.figure(figsize=(14, 10))
plot tree(clf, feature names=features, class names=['No',
'Yes'], filled=True, rounded=True, fontsize=10,
proportion=True)
plt.show()
```



Build KNN Classification model for a given dataset

```
02/04/2025
   Person
                                $2.81
                   50
            18
                                46,57
                  55
            2.3
                                31.95
                  10
            24
    C
                               40.45
     D
            41
                               31.05
                         4
                  70
            2843
                         4
                               60.07
majority dass = 4
   : for X = (35,100)
q. For ivis datuset
  How to choose & value ? Demonstrate
  being acturacy rate and error
   rate.
    Accusey = No. of correct predictions
             Total no. of predictions
   Ex50 5 xate = 1 - accuracy
 I herate through ke or Valker.
 for k in k-values:
      km=knightors amifice (n-neighbou = i)
      kn. fit (x-hain-in's, y-train -iris)
      y- pred = knn . predict (x - test-iris)
      accuracy = accuracy - score (y-test - iris)
      a ceuracies. append ( acuracy)
```

west - rates, append (1 - a ceuracy) choose corresponding k-value with highest mis couse : k=811 1 For diabetes datesset what is the purpose of feature scaling? now to gestorm it? - It is used to separate the dara values from the target variables. y of Fixager is of = dr. drop ('turget', amis = 1) The English of the American The say . 1.30 mm . D *

import numpy as np

```
import pandas as pd
import matplotlib.pyplot as plt
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
from sklearn import datasets
# Load Iris dataset from sklearn
dataset iris = datasets.load diabetes()
X iris = dataset iris.data  # Features
y iris = dataset iris.target # Target
# Split dataset into 80% training and 20% testing
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)
# Choose an appropriate k (e.g., sgrt of number of samples)
k iris = int(np.sqrt(len(X train iris)))
```

```
k iris = k iris if k iris % 2 != 0 else k iris + 1 # Ensure k
# Build and train KNN classifier
knn iris = KNeighborsClassifier(n neighbors=k iris)
knn iris.fit(X train iris, y train iris)
# Predictions and evaluation
y pred iris = knn iris.predict(X test iris)
accuracy iris = accuracy score(y test iris, y pred iris)
conf matrix iris = confusion matrix(y test iris, y pred iris)
report iris = classification report(y test iris, y pred iris)
print(f"Iris Dataset - KNN Classifier (k={k iris})")
print(f"Accuracy: {accuracy iris:.2f}")
print("Confusion Matrix:")
print(conf matrix iris)
print("Classification Report:")
print(report iris)
```

Build Support vector machine model for a given dataset

Screenshot:

```
Points (4:1) 1(4:1) and (0:0) balong to

points clan and points (coliner) and

(o, 1) bits to regards clan . Dres an

opprime hyperplane

$1=(10) $1=(4) $1=(4)

$1=(1) $2=(1) $=(1)

$3:=(1) $2=(1) $=(1)

$3:=(1) $2=(1) $=(1)

$3:=(1) $2=(1) $=(1)

$3:=(1) $2=(1) $=(1)

$3:=(1) $2=(1) $=(1)

$3:=(1) $2=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(1) $3:=(1) $3:=(1)

$3:=(
```

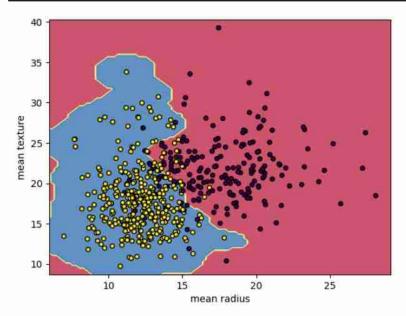
```
# Load the important packages

from sklearn.datasets import load_breast_cancer

import matplotlib.pyplot as plt

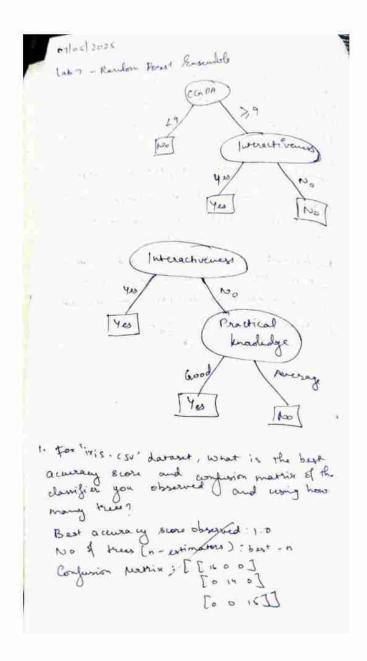
from sklearn.inspection import DecisionBoundaryDisplay
```

```
from sklearn.svm import SVC
# Load the datasets
cancer = load breast cancer()
X = cancer.data[:, :2]
y = cancer.target
#Build the model
svm = SVC(kernel="rbf", gamma=0.5, C=1.0)
# Trained the model
svm.fit(X, y)
# Plot Decision Boundary
DecisionBoundaryDisplay.from_estimator(
        Х,
       response method="predict",
       cmap=plt.cm.Spectral,
       alpha=0.8,
       xlabel=cancer.feature_names[0],
       ylabel=cancer.feature_names[1],
```



Implement Random forest ensemble method on a given dataset

Screenshot:



Code:

import pandas as pd

from sklearn.model_selection import train_test_split

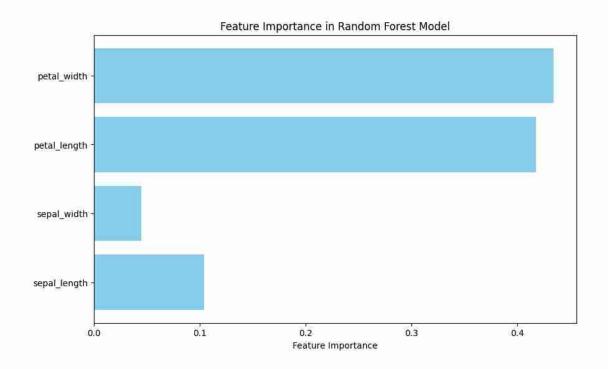
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy score
import matplotlib.pyplot as plt
# Step 1: Load the dataset using pandas read csv
# Replace the file path with the path to your CSV file
data = pd.read csv('/content/iris (6).csv')
# Step 2: Inspect the first few rows of the dataset
print(data.head())
# Step 3: Preprocessing (assuming the target variable is in the
last column)
X = data.iloc[:, :-1] # Feature matrix (all columns except the
target column)
y = data.iloc[:, -1]  # Target variable (last column)
# Step 4: Split the dataset into training and testing sets
X train, X test, y train, y test = train test split(X, y,
test size=0.3, random state=42)
 Step 5: Initialize and train the Random Forest Classifier
```

```
rf model = RandomForestClassifier(n estimators=100,
random state=42)
rf model.fit(X train, y train)
# Step 6: Make predictions on the test set
y pred = rf model.predict(X test)
# Step 7: Evaluate the model performance using accuracy
accuracy = accuracy score(y test, y pred)
print(f"Accuracy of Random Forest model: {accuracy * 100:.2f}%")
# Step 8: Feature Importance (Optional)
feature importances = rf model.feature importances
features = X.columns
# Plotting feature importances
plt.figure(figsize=(10, 6))
plt.barh(features, feature importances, color='skyblue')
plt.xlabel('Feature Importance')
plt.title('Feature Importance in Random Forest Model')
plt.show()
```

```
# Step 9: Hyperparameter Tuning (Optional)
# If you want to perform hyperparameter tuning to optimize the
mode1
from sklearn.model selection import GridSearchCV
param grid = {
    'n estimators': [100, 200, 300],
    'max depth': [None, 10, 20, 30],
    'min samples split': [2, 5, 10],
    'min samples leaf': [1, 2, 4]
# Initialize GridSearchCV
grid search = GridSearchCV(estimator=rf model,
param grid=param grid, cv=5, n jobs=-1, verbose=2)
grid search.fit(X train, y train)
# Print the best parameters and best score
print(f"Best parameters found: {grid search.best params }")
print(f"Best cross-validation score:
{grid search.best score :.2f}")
```

```
# Step 10: Evaluate the model with the best parameters
best_rf_model = grid_search.best_estimator_
y_pred_best = best_rf_model.predict(X_test)

accuracy_best = accuracy_score(y_test, y_pred_best)
print(f"Accuracy of tuned Random Forest model: {accuracy_best * 100:.2f}%")
```



Program 9
Implement Boosting ensemble method on a given dataset

Screenshot:

Sercensilot	
8-Ada Boost Algorithm onlos 2025	but a coursely score and confusion the of the classifier you observed and have
Fredical Actual Weight Sob offer Weight	of the classifier you obscured and has
7 9 No 40 1/L	Law many
7-9 No No 1/2	:16 10 estimators: 0.8277
29 425 425 VL	Conjusion Marine (10 estimators):
75-1	[2138 1406]]
$\mathcal{E}_{\text{CMPM}} = 2 \times \frac{1}{6} = 0.333$	Bust Accuracy: 0.831 with no toth masons.
$\chi_{1600} = \frac{1}{2} \times \ln \left(\frac{1 - 0.333}{0.323} \right)$ $= 0.347$	
7-61PF = 1 x 4 x e + 1 x 2 x e	
= 0-9428 wh(di) = 1 + e -0.247 = 0-1249	
13 (dj) ;+1 = = = + + + 0.0347 = 0-2505	
0.9428	
CGPA Predicted Actual weight	
7=9 425 425 0-2501	
2-9 Yes No 0.2501	
>=9 4es 4es 0.1249 >=9 4es 4es 0.1249	
	- de

Code:

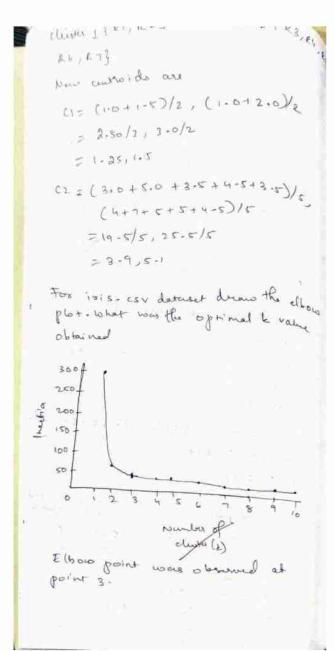
```
import pandas as pd
from sklearn.model selection import train test split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier # For the
base estimator
from sklearn.metrics import accuracy score
# 1. Load the Iris dataset:
# Make sure 'iris.csv' is in the correct location (see
previous response)
data = pd.read csv('/content/income.csv')
# 2. Prepare the data:
X = data.iloc[:, :-1] # Features (all columns except the
last)
y = data.iloc[:, -1]  # Target (last column)
# 3. Split into training and testing sets:
X train, X test, y train, y test = train test split(X, y,
test size=0.3, random state=42)
```

```
4. Create the base estimator (a weak learner):
# We'll use a Decision Tree with a maximum depth of 1 (a
stump)
base estimator = DecisionTreeClassifier(max depth=1)
# 5. Create the AdaBoost classifier:
ada boost = AdaBoostClassifier(estimator=base estimator, #
Using the base estimator
                               n estimators=50,
Number of boosting rounds
                               random state=42)
# 6. Train the model:
ada boost.fit(X train, y train)
# 7. Make predictions:
y pred = ada boost.predict(X test)
# 8. Evaluate the model:
accuracy = accuracy score(y test, y pred)
print(f"Accuracy of AdaBoost on Iris dataset: {accuracy *
98.7:.2f}%")
```

Program 10
Build k-Means algorithm to cluster a set of data stored in a .CSV file

Screenshot:

4 " K Means Algorithm		07/05/2025	
geration I			
guest Co-ordinates	Distance to	Distance to c2	Designal charters
(1.0,1.0)	0.0	7-21	CI
(1-5, 2-0)	1.12	6-12	C +
ps (3.0,4.0)	3.61	3-61	CI
(== 7.0)	7-21	0-0	C 2
10.5.5.0)	4-12	2-5	CZ
14.5.5.0)	5-31	2-06	c 2
R1 (3.5,4.5)	4-10	2.92	62
Charles El, R2, R34	and clust	CITEO.	
pure controids au. CI= (1.0+1.5+3.0) = 5.5/3,7.0/3			"
Now controlde are:)/3, (1.0 [+3.5]/4	r 2-0+4	-o\3
pur controide au. CI = (1.0+1.5+3.0) = 5.5/3,7.0/3 = 1.83,2.33 C2 = (5.0+3.5+4.)/3, (1.0 [+3.5]/4	r 2-0+4	-o\3
Now controlled as: C1 = (1.0+1.5+3.0) = 5.5/3,7.0/3 =1.843,2.33 C2 = (5.0+3.5+4.5) = 16.5/4,21.5 = 4.12,5.87)/3, (1.0 5+3.)/4 7/4	+ 2-0+4 (7+5	-0/3 +5+4-5)/
Now controlle as: CI = (1.0+1.5+3.0) = 5.5/3,7.0/3 = 1.583,2.33 C2 = (5.0+3.5+4.5) = 16.5/4,21.5 = 4.12,5.37 Record Co-ordinate)/3, (1.0 5+3.0)/4 5/4	, (7+5	-0/3 +5+4-5)/
Now controlle as: C1 = (1.0+1.5+3.0) = 5.5/3,7.0/3 = 1.5x3,2.33 C2 = (5.0+3.5+4. = 16.5/4,21.5 = 4.12,5.37 Record Co-ordinate (1.02.05)	1.57	7 2-0+4 (7+5 60 Distant 1002 5-64	-0/3 +5+4-5)/ - Change
Now controlle au. C1 = (1.0+1.5+3.0) = 5.5/3,7.0/3 =1.83,2.33 C2 = (5.0+3.5+4. =16.5/4,21.5 =4.12,5.87 Record Co-ordinate Record (1.0,0)	Dietona 1	7 2-0+4 (7+5) 60 Distant 4007 5:64 4-52	-0/3 +5+4-5)/ - Chans
Pro controide au. C1 = (1.0+1.5+3.0) = 5.5/3,7.0/3 = 1.5x3,2.33 C2 = (5.0+3.5+4. = 16.5/4,21.5 = 4.12,5.37 Record Co-ordinate Record (1.0,0) Record (3.0,4.0)	Dietona	7 2-0+4 (7+5 60 Distant 1002 5-64	-0/3 +5+4-5)/ - Chamo - Chamo
C1 = (1.0+1.5+3.0) = 5.5/3, 7.0/3 = 1.5/3, 2.33 C2 = (5.0+3.5+4.5) = 16.5/4, 21.5 = 4.12, 5.37 Record Co-ordinate (1.0, 20) (3.0,4.0) (4.0,1.0)	Dietoma 1.57 6.41 2.12 5.57	7 2-0+4 (7+5) 60 Distant 4007 5-64 4-52 1-63 1-91	-0/3 +5+4.5/ Arrignal clama () ()
C1 = (1.0+1.5+3.0) = 5.5/3, 7.0/3 = 1.5/3, 2.33 C2 = (5.0+3.5+4.5) = 16.5/4, 21.5 = 4.12, 5.37 Level Co-ordinates C1.5, 2.0) R2 (1.5, 2.0) R3 (3.0,4.0)	Dietona	72-0+4 (7+5 10 Distant 1002 5:64 4:52 1:63	-0/3 +5+4.5)/ Chamber C1 C2 C2

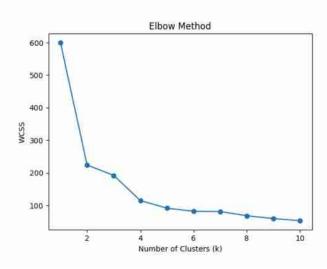


Code:

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
# 1. Load the Iris dataset:
data = pd.read csv('/content/iris (6).csv')
# 2. Prepare the data:
X = data.iloc[:, :-1] # Features (all columns except the
last)
# 3. Standardize the features (important for k-Means):
scaler = StandardScaler()
X scaled = scaler.fit transform(X)
# 4. Determine the optimal number of clusters (using the Elbow
method):
wcss = [] # Within-cluster sum of squares
for i in range(1, 11):
```

```
kmeans = KMeans(n clusters=i, random state=42)
    kmeans.fit(X scaled)
    wcss.append(kmeans.inertia)
# Plot the Elbow method graph
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('WCSS')
plt.show()
# Based on the Elbow method, choose the optimal k (e.g., k=3)
# 5. Apply k-Means clustering with the chosen k:
kmeans = KMeans(n clusters=3, random state=42)
clusters = kmeans.fit predict(X scaled)
# 6. Add the cluster labels to the DataFrame:
data['Cluster'] = clusters
# 7. Visualize the clusters (if possible - 2D or 3D):
```

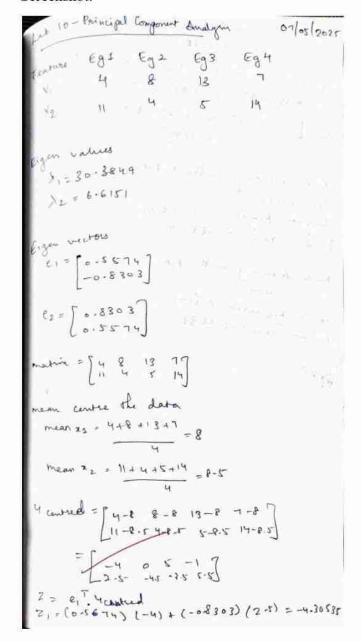
```
Example for 2 features: Sepal Length and Sepal Width
plt.scatter(data['SepalLengthCm'], data['SepalWidthCm'],
c=data['Cluster'], cmap='viridis')
plt.title('k-Means Clustering')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.show()
visualization.
# 8. Analyze the clusters (optional):
# You can examine the characteristics of each cluster by
# calculating the mean values of features within each cluster.
# Example:
 print(data.groupby('Cluster').mean())
```



Program 11

Implement Dimensionality reduction using Principal Component Analysis (PCA) method

Screenshot:



```
would accuracy without PCA
    SVM = 0-8804
  Logistic Regussion: 0-8533
Random Forms: 0-8659
   model Accuracy with PCA
SVM-0.8424
1 Svm - 0.8424
```

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
# 1. Load the Iris dataset:
data = pd.read csv('/content/heart.csv')
# 2. Prepare the data:
X = data.iloc[:, :-1] # Features (all columns except the
last)
y = data.iloc[:, -1]  # Target variable (last column)
# 3. Standardize the features (important for PCA):
scaler = StandardScaler()
X scaled = scaler.fit transform(X)
# 4. Apply PCA:
# Choose the number of components (n components) you want to
keep
# For example, to reduce to 2 dimensions:
```

```
pca = PCA(n components=2)
X pca = pca.fit transform(X scaled)
# 5. Create a new DataFrame with the reduced dimensions:
pca df = pd.DataFrame(data=X pca, columns=['PC1', 'PC2'])
pca df['Species'] = y # Add the target variable back for
visualization
# 6. Visualize the reduced data:
plt.figure(figsize=(8, 6))
plt.scatter(pca df['PC1'], pca df['PC2'],
c=pca df['Species'].astype('category').cat.codes,
cmap='viridis')
plt.title('PCA of HeartDataset')
plt.xlabel('Principal Component 1 (PC1)')
plt.ylabel('Principal Component 2 (PC2)')
plt.show()
# 7. Explained variance ratio:
# This tells you how much variance is explained by each
principal component
explained variance ratio = pca.explained variance ratio
```

print(f"Explained Variance Ratio: {explained_variance_ratio}")

