# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.



## LAB REPORT on

## **Machine Learning (23CS6PCMAL)**

Submitted by

Anjali (1BM22CS042)

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 **Anjali (1BM22CS042),** 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.

Saritha A. N	Dr. Kavitha Sooda
Assistant Professor	Professor & HOD
Department of CSE, BMSCE	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	4
2	3-3-2025	Demonstrate various data pre-processing techniques for a given dataset	7
3	10-3-2025	Implement Linear and Multi-Linear Regression algorithm using appropriate dataset	13
4	17-3-2025	Build Logistic Regression Model for a given dataset	24
5	24-3-2025	Use an appropriate data set for building the decision tree (ID3) and apply this knowledge to classify a new sample.	28
6	7-4-2025	Build KNN Classification model for a given dataset.	33
7	21-4-2025	Build Support vector machine model for a given dataset	45
8	5-5-2025	Implement Random forest ensemble method on a given dataset.	53
9	5-5-2025	Implement Boosting ensemble method on a given dataset.	56
10	12-5-2025	Build k-Means algorithm to cluster a set of data stored in a .CSV file.	59
11	12-5-2025	Implement Dimensionality reduction using Principal Component Analysis (PCA) method.	65

Github Link: https://github.com/Anjali-042/ML\_LAB

#### Program 1

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

_1-1-1-1	Committee LAB-1 was abstract to the last	100	Date
Company of the	min - my co when the min	-	Poge
#_	Mouling.csv		longitude bongitude howing median age Total noom Istal
cio	import pandas as pd	ciii	lengitude langutude houring-many
0.0	of = pd. reades _ csv ( '/ Content / 1553 7 68847 - howing.	nin_	- 194. 35 32.5 <del>4</del>
elia	privit ( Sample data )		appulation households median-income median-house volus
	print (di head ())		population households median - microme median hour volus
	print (d. head ()) print ("")	U	(0.00)
	A E 17 COLOR OLD A		Calaba (Marks) Bang
_min	ALL COLOR CO	CIVI	ocean proximity
_cii>	print (df. info ())		614 Ocean 3136
	A STATE OF THE PARTY OF THE PAR		(1) Ocean 9136 INLAND 6551
_ ciii)	print (of describe ())	4	Near Ocean 2658
(iv)	print (d) [ 'ouan - proximity '] value counts ()		Near bay 2290
CIVI	power of the state		Island 5
	Off -		as mind all 1 so year allowed to the allowed to
La	i ti or tital normal	100	print (of inul () . wm())
ciis	longitude latitude howing oudian age total nooms	(V)	February 19
	- 122B 880		total-bedrooms 207
	-122-3	2000	John see milion process with the state
			0 18 PP 88.68 8.3314
	129.0 126 8.3352	- 3	Trendston to allow the trendstone
	1	-	100 191 9 3260 Ness Cut
	median-howing-value		6
	median-houring value 452600		anton paring walnut
			July (ST
= Di	abeter and Adult incomo		
is pr	int (of isnul c) - um ()		
a. 1N	o data column had runing values in diabeter dataset	-	
a.	nd income dataset		
		1	
iis G	render is a categorical column		
1	print (of info ())		
	1 1213		
	the to a make same,		
iii)	Nin - max Scaling reales feature to a specific range, typically between 0 and and stordardization centure typically between 0 and and stordard desiration of	į.	
1	typically trueen o and and standardization centers	1	
	The data around zero with the		
	+ + mainta	10	
	Min-max usling is und when we want to the original data distribution and standardization is the original data distribution and standardization is	i	
	the original data distribution and somewhat of suttiness	8	
		ł	
The state of	202 Little amount 1. Mar 180		
trib link	and when the state of the state		
A S A S A S A S A S A S A S A S A S A S	or tout stant a hind to		
	The affiliation of the	1	
	Margh = " Sample of		
	at = pd. sund, evy 1 (1)	1	
	College and stop stopped I truly	1	
	Parket Synother &	Ī	
	o point siles sint themes of point o	Ī	

```
Code:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model selection import train test split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from scipy import stats
#**Diabetes Dataset**
df=pd.read_csv('/content/Dataset of Diabetes .csv')
df.head()
df.shape
print(df.info())
# Summary statistics
print(df.describe())
missing values=df.isnull().sum()
print(missing_values[missing_values > 0])
categorical cols = df.select dtypes(include=['object']).columns
print("Categorical columns identified:", categorical cols)
if len(categorical_cols) > 0:
  df = pd.get_dummies(df, columns=categorical_cols, drop_first=True)
  print("\nDataFrame after one-hot encoding:")
  print(df.head())
else:
  print("\nNo categorical columns found in the dataset.")
from sklearn.preprocessing import MinMaxScaler, StandardScaler
import pandas as pd
numerical_cols = df.select_dtypes(include=['number']).columns
scaler = MinMaxScaler()
df_minmax = df.copy() # Create a copy to avoid modifying the original
df minmax[numerical cols] = scaler.fit transform(df[numerical cols])
scaler = StandardScaler()
df standard = df.copy()
df standard[numerical_cols] = scaler.fit_transform(df[numerical_cols])
print("\nDataFrame after Min-Max Scaling:")
print(df_minmax.head())
print("\nDataFrame after Standardization:")
print(df_standard.head())
#**Adult Income Dataset**
df1=pd.read_csv('/content/adult.csv')
df1.head()
df1.shape
```

```
print(df1.info())
# Summary statistics
print(df.describe())
missing_values=df1.isnull().sum()
print(missing values[missing values > 0])
categorical_cols = df1.select_dtypes(include=['object']).columns
print("Categorical columns identified:", categorical_cols)
if len(categorical cols) > 0:
  df1 = pd.get_dummies(df1, columns=categorical_cols, drop_first=True)
  print("\nDataFrame after one-hot encoding:")
  print(df.head())
else:
  print("\nNo categorical columns found in the dataset.")
from sklearn.preprocessing import MinMaxScaler, StandardScaler
import pandas as pd
numerical_cols = df1.select_dtypes(include=['number']).columns
scaler = MinMaxScaler()
df minmax = df1.copy() # Create a copy to avoid modifying the original
df_minmax[numerical_cols] = scaler.fit_transform(df1[numerical_cols])
scaler = StandardScaler()
df_standard = df1.copy()
df_standard[numerical_cols] = scaler.fit_transform(df1[numerical_cols])
print("\nDataFrame after Min-Max Scaling:")
print(df_minmax.head())
print("\nDataFrame after Standardization:")
print(df_standard.head())
```

## PROGRAM 2 Demonstrate various data pre-processing techniques for a given dataset Screenshot

Date	+. df. to-csv (sate +
1AB-1	print (" Data saved to content csv")
import pandas as pd	-> tickurs = [ 'NDEGRANKINS", ICICI BANKINS, "RO ARB ANKINS"] data = W. desmond (+ 1)
dA = ("USN": [1,2,3,4,5), "Name": [A,8,5,4]	data = 15 desemberd (tubine, start = "2024-01-01"
	data = 4f. desembed (tubins, start = "2024-01-01")  lind = "2024-12-30", group-by = "ticknos")  import ufinance as 4f. import pandax as pd  import matplotlib. pyplot as plt
df = pd.dataframe(df) print(df)	-> hase - data ( 'HOFCBANK, NS , ]
O I A 10	nage - data [ Daily - Relian ] = hage-dato [ close ]
2 B 20	point ("Raily Return for MDF(") pot.ch point (hafe-data ("Daily return"] head (")
3 4 0 40	plt. Jigure ( figs12e = (0,6))
4 15 m E de 150 parte de la	ndfe data [ close ]. plot (Title = 1. NDFC Indutaria
from skleann datauts impart load - diahetes diahetis = load - diahetis ()	01+ (1) dat (2,1,2) 01 + 0 = x1x
of = pd. Pataframe (diabeter data, columns = diabeter data, columns = diabeter tanget	plt. subplot (2014) notion ). plot (title = "NDFC I - Daily notion =  plt. hight layout ()  off. show ()
dg-head ()	plt-hight layout ()
file path = 'Sample - Saler - data - 15V'	ot show ()
of = pd. read - csv (file-path) print (sample data)	2024-01-01 NaN 2029-01-02 -0.015420
	2024-01-03 -0.015420
Produit Quaritity Price Sales Region 1 1aptop 5 1000 5000 North	

```
Code
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
df=pd.read_csv('housing.csv')
df.head(2)
df.describe()
df.info()
sns.histplot(df['median_income'], kde=True, color='green')
sns.histplot(df['housing_median_age'])
from sklearn.model_selection import train_test_split
X = df.drop("median_house_value", axis=1)
y = df["median_house_value"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=42)
X = df.drop("median_house_value", axis=1)
y = df["median_house_value"]
df["income_cat"] = pd.cut(df["median_house_value"],
bins=[0, 100000, 200000, 300000, 400000, np.inf],
labels=[1, 2, 3, 4, 5])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42,
stratify=df["income_cat"])
```

```
train_set = X_train.copy()
train set["median house value"] = y train
train_set.plot(kind="scatter", x="longitude", y="latitude", alpha=0.4,s=train_set["population"]/100,
label="population",figsize=(10,7), c="median_house_value", cmap=plt.get_cmap("jet"),
colorbar=True)
plt.legend()
numerical_columns = df.select_dtypes(include=['float64', 'int64'])
correlation_matrix = numerical_columns.corr()
print(correlation matrix["median house value"].sort values(ascending=False))
df.plot(kind="scatter", x="median income", y="median house value", alpha=0.1)
# Combine 'median income' and 'households'
df["income_households"] = df["median_income"] * df["households"]
numerical_columns = df.select_dtypes(include=['float64', 'int64'])
correlation matrix = numerical columns.corr()
print(correlation matrix["median house value"].sort values(ascending=False))
df.plot(kind="scatter", x="income_households", y="median_house_value", alpha=0.1)
plt.show()
missing_values = df.isnull().sum()
print(missing values[missing values > 0])
h=df
h.dropna(subset=["total_bedrooms"])
from sklearn.preprocessing import OneHotEncoder
df1=pd.read_csv('housing.csv')
hc=df1[["ocean_proximity"]]
```

```
encoder=OneHotEncoder()
hc_encoded=encoder.fit_transform(hc).toarray()
hc_1hot_df = pd.DataFrame(hc_encoded, columns=encoder.get_feature_names_out(hc.columns))
hc_1hot_df.head()
Feature scaling is crucial in machine learning for several reasons, particularly when using algorithms that
are sensitive to the scale of features. Here's a breakdown of its importance:
1. Improved Performance of Distance-Based Algorithms:
2. Faster Convergence of Gradient Descent:
3. Improved Regularization:
4. Better Interpretation of Coefficients:
5. Numerical Stability:
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
# Custom transformer to add engineered attributes
class CombinedAttributesAdder(BaseEstimator, TransformerMixin):
  def __init__(self, add_bedrooms_per_room=True):
```

```
self.add_bedrooms_per_room = add_bedrooms_per_room
  def fit(self, X, y=None):
    return self
  def transform(self, X):
    # Assumes X is a NumPy array with the following columns:
    # total_rooms (index 3), total_bedrooms (index 2), population (index 4), households (index 5)
    rooms_per_household = X[:, 3] / X[:, 5]
    population_per_household = X[:, 4] / X[:, 5]
    if self.add_bedrooms_per_room:
       bedrooms_per_room = X[:, 2] / X[:, 3]
       return np.c_[X, rooms_per_household, population_per_household, bedrooms_per_room]
    else:
       return np.c_[X, rooms_per_household, population_per_household]
# Identify numerical and categorical columns
num_attribs = df1.drop("ocean_proximity", axis=1).columns # All numeric columns
cat_attribs = ["ocean_proximity"]
# Build numerical pipeline: impute missing values, add new attributes, then scale
num_pipeline = Pipeline([
  ('imputer', SimpleImputer(strategy="median")),
  ('attribs_adder', CombinedAttributesAdder()),
  ('std_scaler', StandardScaler()),
```

```
# Build the full pipeline combining numerical and categorical processing
full_pipeline = ColumnTransformer([
    ("num", num_pipeline, num_attribs),
    ("cat", OneHotEncoder(), cat_attribs),

# Process the dataset using the pipeline
housing_prepared = full_pipeline.fit_transform(housing)
print("Shape of processed data:", housing_prepared.shape)
```

PROGRAM 3 Implement Linear and Multi-Linear Regression algorithm using appropriate dataset Screenshot

	Drite Proge
One	
	$((x^Tx)^{-1}x^T)Y = [1.0 - 0.5 0 - 0.5]$
10b - 4	-0.3 -0.1 0.1 0.3 4
lab - 4	5 Lynn Regular
Linear Regression	(1)
	= [-0.5] by an arbitrar tragen
week xi Salu yi	2.2 an in uponer tragen
1 2	to in talque, ditribution Termal
" in-in-12 now = both, intil hoursest in - all	from dream of the party of the
( 5003 181 = 100 gum 5 19 As - 21 - 10 10 12 - 1	Sum others and relation jugant than that gelit
by in Ashin Paris 912 in remain time	11 = a + a x + a 2 2
1 the 10 It long Lateration than	$y = a_0 + a_1 x_1 * a_2 * a_2$
$x^{T} = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$	0 - 5
THE HATE CHAPTERANKING ? ? .	
4T = 2 4 5 9 1 1 1 1 1	y=-0.5+(2.2)
to to	
and commended the supplier of	H. Colimpie = The Colimpie to the
X = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	*
1 2 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9	* steps (algo)  import ubnaries  import data distribution  data distribution
1 3	of the state of th
1 3	3 D impart libraries
The state of the s	2) import transition 2) import data - distribution 3) Analyse data - distribution 3) Analyse data - distribution
$x^Tx = 4$ 10 cm taple to	2) Import data - distribution 3) Analyse data - distribution 4) Outsubstiem plot - visualisation 4) Outsubstiem plot - visualisation
$X^TX = 4$ 10 cm taplax to	37 Amalyx dot - villatisalism
10, 30 dian and yet	2) Quitabutier turiable
- acts & country which	Doctionship blu mark
(xTx) = 1.5 -0.5 00 taged to	5) With the data
-0.5 0.2	3) Analyse dala - duminimusisation 4) Quitabition plot - virualisation 5) Reationship b [w variable 5) Split the data 6) Split the data 7) Train the model 9) Predict the result
Y CAN A STATE OF THE STATE OF T	
(xTx)-1 x T = may 1.0 0.5 1000 -008	Oudit the state
(xTx) x = 100 1.0 0.1 0.3	8) his oredition and intercept
MCN310-5-10-3 0.1 08.	a) Virialice o coefficient and
1 4 CHILD:0 - 180-1-100	g) Vinalize predutation and intercept
	9) Predict the regult  9) Predict the regult  10) Values of Coefficient and interest  10) Values of Coefficient and interest
	1
	(a) Values of Contract of Cont
	The state of the s
	A STATE OF THE PARTY OF THE PAR

-	linear Pressurem	I	Poge
P	Limean Regnession		1. O/P - manupatornia - laboral
	import pandas as pd	1	Predicted per capita income 2020: \$ 11288-69
		1	
	import numpy as no import matplotub, pyplot as pit	2	
	Con della della della inserta train to		preduted salary for 12 years of experience: \$139093.67
	from skleam model relation impart train to	a splat	# Multiple Regression
-	from skleam linear model import linear regress	uen =	import pandas as od
-	from sklean metric import mean-squared en	non	imposit num py as no
			import matphotis, puplot as alt
	Sile path = " (content ) (anada - pon - capita inc	came. csv	from sklearn linear model import linear Regression
	df=near		
	de = pd. read_csv (file-path)	11 2	file-path = "   Content   hiring. Esv"
	df. columns = ["year", "per_ capita income"	+3	df = pd. read _ csv (file_path)
			to tendered solven for foundation 1: 3 95001-11
	X = df [["year"]]	chan a	of loc [: , "text_wore (out of 10)"] = of ["text_wore(
	Y = of [" pon_apita_income "]	Action of	out of 10)"]. sillpac of ["tet-use ("out of 10")].m
	model = linear Regrusion ()	Inga G	BE DESUIS & TAIL MANNING . 7
	model (+ (X, X)		def Convert experience (val):
	model - gat (X, 1)	1/2×3 (2)	t control to
	predicted income 2000 = model predict [[	(2020]]	2000 = 1 10000 = 1 2000 " O THE . 1 , WALL
	preducted - vices in - labora tal in	DUT. (E	more 13, on
	plt. scatter (x, y, color = 'blue', label = 'Acc	dual Data?	return rised to num. get (val Journ (), ())
	plt. scatter (x, y, color = blue, late = blue, late plt. plot (x, made) predict(x), color = blue; late plt.	abel = Reg	peturn val
	at olat (x, model-place	miles line	- Contraction of the Contraction
	plt. Mahd ("year")		of loc [:, "expenione"] = of ["experione"]. apply (
_	ptt. ylabel ("pen apita income")	N. Ball	of loc [:, experience convert experien
_	pt. legend ()		of drappia (implace = True) " " tod score (out of 10)",
			dg. dropna (implace = lace)  x = ag [I "experience", "tid score(ant og 10)",  x = ag [I "experience", "tid score(ant og 10)"]
	Olto show ()	r S apredicted	x = df [ " explusion 1 1 + 41 10 )" ]?
_			10000 000
	1 1 1 1 1 000 noita mismo 2020:	PHP	" intinuer - more Cour of to
	print (+ "Pridited per capita incomp 2020: 1  print (+ "Pridited per	PHP	y = of [" salary (\$)?]  y = of [" salary (\$)?]
P. P. O.S.	model = linear Requision  model = git (x, y)  Condidate = pd. Data from ((f 2, 3, 6), [12, 10, 10])  Columns = ("experience", " mit was (out of 10)")  Jan i, solary in sommation (predicted solaries);  print ( + " Enduted solary Jan Condidate (1+1); }  Solary : 2+1 (2)  Of P	3. Predicted	Salary: \$ 26,000  meoded catigorial value like state wing one with the catigorial value into binary
I. Iran	model = Linear Requision  (authorized (authorized (authorized))  for it is clearly in tenuminate (preduited salarised);  paint( + " Preduited Salary for Condidate (1+1);  foolige = Condidate   1   3   3   3   3   3    Preduited Salary for Condidate   3   3   3   3   3    Preduited Salary for Condidate   3   3   3   3   3    Preduited Salary for Condidate   3   3   3   3   3    Preduited Salary for Condidate   3   3    Pr	3. Enchited  Yes., A confirmation of the second of the sec	Salary: \$ 36,000  medded catigorial values like state wing one wat g to comment the catigorial values into binary
1	model = Linear Regulier  model = Linear Regulier  model = Git (x, y)  Cardidates = pd. Data from (C(C 2, 3, 67, [12, 10, 10])  Cardidates = pd. Data from (C(C 2, 3, 67, [12, 10, 10])  Cardidates = pd. Data from (C(C 2, 3, 67, [12, 10, 10])  Cardidates = pd. Data from (C(C 2, 3, 67, [12, 10, 10])  Columns = ("exposions", "athlean (cut of 10)")  Jan i, solary in commonate (predited solarius):  print (f * Condited Solary for Condidate 1 : 4 38581.11  Colp =  Condited Solary for Condidate 1 : 4 38581.11  Predited Solary for Condidate 2 : 4 56486.23  Condited Color for Condidate 2 : 4 56486.23	3. Enchited  Yes., A confirmation of the second of the sec	Salary: \$ 36,000  medeld catigorial value like state wing one ut g to comment the catigorial values into binary
1	model = Linear Reguisser  (at least of 10) "  (churing = ("exposiones", "tat uses (suit of 10)"  for i, solvey in triuminate (predicted solvey);  print( + "Predicted Solvey for Condidate (1+1);  (solvey for Condidate 1: \$ 38581.11  Predicted Solvey for Condidate 2: \$ 53486.23  Rudicted Roefit: \$ 510570.99	3. Enchited  Yes., A confirmation of the second of the sec	Salary: \$ 36,000  medeld catigorial value like state wing one wat g to comment the catigorial values into binary
I.	model = linear Reguisson  (consider = pd. Ontafrance ((cf. 2, 3, 6), [12, 10, 10])  columns = ("exposions", "model solor)  for i, solary in soluminate (pacitid solor)  point ( + "ondited solory for condidate (1+1); }  solary : 2+19,  Olp =  Condited solary for condidate 1: \$ 38581.11  Dudited solary for condidate 2: \$ 56486.23  Radited Profit: \$ 5105.70.83	3. Enchited  Yes., A confirmation of the second of the sec	Salary: \$ 36,000  medeld catigorial value like state wing one wat g to comment the catigorial values into binary
, little	model = Linear Regardier  (ext of 10) "  for i , relaxy in remandate (predicted solvaise):  parial ( = "Credicted Solvay for landidate (1+1): 1  ( solvay = 2+7")  Colp = Credicted Solvay for Condidate 2: 458486.83  Reducted Reefit: \$ 510870.93	3. Enchited  Yes, A to Encoding  yaniable	Salvey: \$ 26,000  meocled catigorial value like state wing one with the catigorial values into binary
, little	model = Linear Regardier  (ext of 10) "  for i , relaxy in remandate (predicted solvaise):  parial ( = "Credicted Solvay for landidate (1+1): 1  ( solvay = 2+7")  Colp = Credicted Solvay for Condidate 2: 458486.83  Reducted Reefit: \$ 510870.93	3. Enchited  Yes, A to Encoding  yaniable	Salvey: \$ 26,000  meocled catigorial value like state wing one with the catigorial values into binary
, little	model = Linear Regardier  (ext of 10) "  for i , relaxy in remandate (predicted solvaise):  parial ( = "Credicted Solvay for landidate (1+1): 1  ( solvay = 2+7")  Colp = Credicted Solvay for Condidate 2: 458486.83  Reducted Reefit: \$ 510870.93	3. Enchited  Yes, A to Encoding  yaniable	Salary: \$ 26,000  meocled catigorial value like state wing one with the catigorial value into binary
, little	model = linear Reguisson  (consider = pd. Ontafrance (C. (2,3,6), [12,10,10])  columns = ("exposions", "model solor)  for i, solary in somewhole (partited solorius);  paint ( + "ondited solory for condidate (1+1); }  Solary : 2+19, )  Olp =  Condited solory for condidate 1: \$ 38581.11  Dudited solory for condidate 2: \$ 55486.23  Radited Profit: \$ 5105.70.83	3 Enchited  Xes, A se  Firedony  yaniable	Salary: \$ 26,000  meocled catigorial value like state wing one with the catigorial value into binary  The consumption of the catigorial value into binary  The catigorial value into binary  Th
ISO	model = Linear Regulier  model = Linear Regulier  model = Jit (x, y)  Condidates = pd. Data from ((((2,2,6), [12,10,10]))  Condidates = pd. Data from ((((2,2,6), [12,10,10]))  Columns = ("Expositions", " and wave (out of 10)")  Jan i, was y in unuminate (predicted solvains):  private f = Predicted solvany for landidate (1);  (solvany = 2+1, 2)  Producted Solvany for (andidate 1: \$38581.11  Producted Solvany for (andidate 2: \$55486.23)  Reducted Profit: \$510590.99  Res of preformed data provising stops like Handling mining values and exceeding entegorical data to Insurate data and ready for modeling.	3 Enchited  Xes, A se  Firedony  yaniable	Salary: \$ 26,000  meocled catigorial value like state using one ut g to consuct the catigorial values into binary
150	model = Linear Regulium  (cat of 12, 12, 10, 10)  Jan i, Iclary in Insumunat (predited salarium):  print (	3 Enchited  Xes, A se  Firedony  yaniable	Salary: \$ 26,000  meocled catigorial value like state wing one with the catigorial value into binary  The consumption of the catigorial value into binary  The catigorial value into binary  Th
15	model = linear Reguission  model = linear Reguission  model = git (x, y)  consider = pd. Ontagrame ((f 2, 3, 6), [12, 10, 10])  columns = ("experiens", " mit uses (out of 10)")  for i, solary in summont (predicted solaries);  point ( + "Ordited solary for condidate \$1+13. 1  \$ solary : 2+19.)  Ole =  Condited solary for condidate 1: \$ 38581.11  Dudited solary for condidate 2: \$ 55486.23  Radited Profit: \$ 5105.70.83  Residuted Profit: \$ 5105.70.83  Residuted Profit: \$ 100.70.83  Residuted Profit: \$ 5105.70.83  The solaries and encoding categorisal data to Invest the datasets on Jean and ready for modeling.  The regression line is slopes uppeared, indicating that are contral encourse impressed auto time.	3. Enchited  Yes, A constitution of the consti	Salary: \$ 26,000  medial catigorial value like state wing one wat g to comment the catigorial value into kinary  Constant to the catigorial value into kinary  C
, , , , , , , , , , , , , , , , , , , ,	model = Linear Regulium  (cat of 12, 12, 10, 10)  Jan i, Iclary in Insumunat (predited salarium):  print (	3. Enclisted  Yes., A war aniable.	Salary: \$ 26,000  medded catigorial value like state wing one wat g to comment the catigorial values into kinary  Constitution of the catigorial values into ki
, , , , , , , , , , , , , , , , , , ,	model = linear Reguission  model = linear Reguission  model = git (x, y)  consider = pd. Ontagrame ((f 2, 3, 6), [12, 10, 10])  columns = ("experiens", " mit uses (out of 10)")  for i, solary in summont (predicted solaries);  point ( + "Ordited solary for condidate \$1+13. 1  \$ solary : 2+19.)  Ole =  Condited solary for condidate 1: \$ 38581.11  Dudited solary for condidate 2: \$ 55486.23  Radited Profit: \$ 5105.70.83  Residuted Profit: \$ 5105.70.83  Residuted Profit: \$ 100.70.83  Residuted Profit: \$ 5105.70.83  The solaries and encoding categorisal data to Invest the datasets on Jean and ready for modeling.  The regression line is slopes uppeared, indicating that are contral encourse impressed auto time.	3. Enclisted  Yes., A war aniable.	Salary: \$ 26,000  medial catigorial value like state wing one wat g to comment the catigorial value into kinary  Constant to the catigorial value into kinary  C

```
Code
# -*- coding: utf-8 -*-
import pandas as pd
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
df = pd.read_csv('/content/housing_area_price.csv')
df
# Commented out IPython magic to ensure Python compatibility.
# %matplotlib inline
plt.xlabel('area')
plt.ylabel('price')
plt.scatter(df.area,df.price,color='red',marker='+')
new_df = df.drop('price',axis='columns')
new_df
price = df.price
price
# Create linear regression object
reg = linear_model.LinearRegression()
reg.fit(new_df,price)
```

```
"""(1) Predict price of a home with area = 3300 sqr ft"""
reg.predict([[3300]])
reg.coef_
reg.intercept_
"""Y = m * X + b (m is coefficient and b is intercept)"""
3300*135.78767123 + 180616.43835616432
"""(1) Predict price of a home with area = 5000 sqr ft"""
reg.predict([[5000]])
# -*- coding: utf-8 -*-
import pandas as pd
import numpy as np
from sklearn import linear_model
df = pd.read_csv('/content/homeprices_Multiple_LR.csv')
df
```

```
"""Data Preprocessing: Fill NA values with median value of a column"""
df.bedrooms.median()
df.bedrooms = df.bedrooms.fillna(df.bedrooms.median())
df
reg = linear_model.LinearRegression()
reg.fit(df.drop('price',axis='columns'),df.price)
reg.coef_
reg.intercept_
"""Find price of home with 3000 sqr ft area, 3 bedrooms, 40 year old"""
reg.predict([[3000, 3, 40]])
112.06244194*3000 + 23388.88007794*3 + -3231.71790863*40 + 221323.00186540384
import pandas as pd
from sklearn.linear_model import LinearRegression
# Load the dataset
df1 = pd.read_csv('/content/canada_per_capita_income.csv')
```

```
# Prepare the data
X = df1.year.values.reshape(-1, 1) # Features (year)
y = df1['per capita income (US\$)'] # Target (per capita income)
# Create and train the linear regression model
model = LinearRegression()
model.fit(X, y)
# Predict per capita income for 2020
year_2020 = [[2020]]
predicted_income = model.predict(year_2020)
print(f"Predicted per capita income for Canada in 2020: {predicted_income[0]:.2f}")
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
# Load the dataset (canada_per_capita_income.csv)
df1 = pd.read_csv('/content/canada_per_capita_income.csv')
# Prepare the data
X = df1.year.values.reshape(-1, 1) # Features (year)
```

```
y = df1['per capita income (US$)'] # Target (per capita income)
# Create and train the linear regression model
model = LinearRegression()
model.fit(X, y)
# Create the plot
plt.figure(figsize=(8, 6))
plt.scatter(X, y, color='blue', label='Data Points') # Now using the correct X and y
plt.plot(X, model.predict(X), color='red', label='Regression Line')
plt.xlabel('Year')
plt.ylabel('Per Capita Income (US$)')
plt.title('Per Capita Income in Canada over Time')
plt.legend()
plt.grid(True)
plt.show()
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.impute import SimpleImputer
# Load the dataset
df = pd.read_csv('/content/salary.csv')
# Prepare the data
```

```
X = df.iloc[:, :-1].values # Features (years of experience)
y = df.iloc[:, 1].values # Target (salary)
# Impute missing values with the mean
imputer = SimpleImputer(strategy='mean') # Create an imputer object with strategy as mean
X = imputer.fit_transform(X) # Fit and transform the imputer on feature data 'X'
# Create and train the linear regression model
model = LinearRegression()
model.fit(X, y)
# Predict salary for 12 years of experience
years_experience = [[12]]
predicted_salary = model.predict(years_experience)
print(f"Predicted salary for 12 years of experience: {predicted_salary[0]:.2f}")
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.impute import SimpleImputer
# Load the dataset
df = pd.read_csv('/content/hiring.csv')
# Handle missing values
```

```
# Convert 'experience' column to numeric, replacing non-numeric with NaN
df['experience'] = pd.to_numeric(df['experience'], errors='coerce')
imputer = SimpleImputer(strategy='mean')
df['experience'] = imputer.fit_transform(df[['experience']])
df['test_score(out of 10)'] = imputer.fit_transform(df[['test_score(out of 10)']])
# Prepare the data
X = df.drop('salary($)', axis='columns')
y = df['salary(\$)']
# Create and train the linear regression model
model = LinearRegression()
model.fit(X, y)
# Predict salaries for the given candidates
candidate1 = [[2, 9, 6]]
candidate2 = [[12, 10, 10]]
predicted_salary1 = model.predict(candidate1)
predicted_salary2 = model.predict(candidate2)
print(f"Predicted salary for candidate 1: ${predicted_salary1[0]:.2f}")
print(f"Predicted salary for candidate 2: ${predicted_salary2[0]:.2f}")
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer
# Load the dataset
df = pd.read_csv('/content/1000_Companies.csv')
# Separate features (X) and target (y)
X = df.iloc[:, :-1].values
y = df.iloc[:, 4].values
# Encode categorical data (State)
labelencoder = LabelEncoder()
X[:, 3] = labelencoder.fit\_transform(X[:, 3])
ct = ColumnTransformer(
  transformers=[('encoder', OneHotEncoder(), [3])],
  remainder='passthrough'
)
X = \text{ct.fit\_transform}(X)
# Avoid dummy variable trap (remove one encoded column)
```

```
X = X[:, 1:]

# Split data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Create and train the multiple linear regression model
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predict profit for the given values
new_prediction = regressor.predict([[1, 0, 91694.48, 515841.3, 11931.24]])
```

print(f"Predicted Profit: {new\_prediction[0]:.2f}")

#### PROGRAM 4 Build Logistic Regression Model for a given dataset

lab = 3:	
	Date
hiven $a_0 = -5$ $a_1 = 0.8$	2 N - dol
which to remain the contraction where printer things of	SATTMON (7.) = e2 10 50 0.466 World
1) logitu regression equation	Softmax $(z_1) = e^2$ 0.465 mm
	Impart pandas as ad
$p(x) = \frac{1}{1 + e^{-(a_0 + d_1 bx)}} = \frac{1}{1 + e^{-(-5 + 0 \pi x)}}$	Softmax (22) = e' = 0 244
1+e-(a0+a,00) 1+e-(-5+0.80)	e2+e1+e02 how 19 h
- Maria Cara Anna Cara Cara Cara Cara Cara Cara Cara C	At know co
i) Cale protability that a student who studies	Softmax $(z_3) = e^2 + e^1 + e^2$
7 how will pass	e 2+ e'+ e 0
	tips to viet trans nother was the tit of it
$\rightarrow x = 7 \rho(x) = 1$	probability of the 3 dairs are appar 10.54, 24.4
1+e-(-5+0.8(7))	1/2 and 31/2
= 0.6457	90 6 33
while haled the suret or a second	Loun Sheen William W.
ciii) Determine the predicted class (PIF) for this two	from Edward Linear moch grapest segetter Regression
based on thosehold at a 5	Court want of the house
0.5	X. This
-> PCx) = 0.6457	hat y
	Chil - Thilling - made postile ( + 1)
Thus y = 1 (pan) 0 otherwise	has like it
Life and the second of the second	tande blue
2) (midu 2 = [0 1 0] 1 4 4	the state of the s
2) Cerricles Z = [2,1,0] for some classes apply so to find probabilities values of 3 classes.	Many tompin
to find probabilities values of 3 claves.	Lanna lab
/ 0	(( ) ma. them +1) } t willing
$Softmax (21) = e^{2x}$	The state of the s
Softmax (212) = $e^{2ix}$ $\mathcal{E}_{j=1}^{i}$ $e^{2j}$	Cart was marked to
	248 2007 4500 = 5
	Con Structure I
	( u Yamile

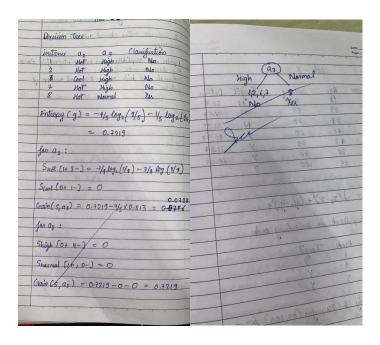
100-13	buse 11
Binary logitic Regression	09l = 35
	age = 35 prediction_function (age)
import pandar as pd	gumas (age)
From matplottib impart pypist as pit	0/9 - 0.3703834
of pd. mad - ssy ("Tronton )	
d. had ()	Laborary on the Treat of the Co.
of hand ()  pot realter (of age, of bought invunance, marken = 4)	Muticlass logitic Regression
	1.1.1
y tours y teil u tours, he had a main ten splat of	import pandas as pd
D, df. bought invinance, Train-size = 0.9 , random state = 10	from sklearn datauti import load inic
	from sklearn model - relation import train tot - split
x-Tit	gram sklearn linear model impart lagithe Regression
from sthoon Linear model import logatic Regression model = logatic Rigression ()	from sklean metric impart accuracy score
model = logitic Regression ()	1340 from skelearm import metrics
model sit (x train, y train)	
X_Tut	inix = pd nead -cov-("/Content/inix.cov")
y_tid	The transfer of the country of the c
x-predicted = model. predict (x-ted)	X = inix doop ("species", aris = "columna")
y - predicted	y = iris species x to in x test = train ted split
model roef-	y = iris, species x train, x-trid, x train, x trid = train Trid = split x train, x-trid, x train, x trid = train Trid = split
model_intercept_	x_train, x_tut, Y_train, Y_tut = noin to x_y, tut_size = 0.2, nander= stit
	the residence of the transfer of the contract of
import math	in Coulti clau = multinomia
def rignaid (2):	
return 1 (C+ math. exp(-x))	
	- model pridit (x-tul)
def predation-func (age): z = 0.12 + 1 age - 4.8+3	y pard = model predict (x tul)  y pard = model predict (x tul)  aurocy = aurocog - were (y tul, y - pred)  aurocy = aurocog - were (y tul, y - pred)
Z = 0.127 * age - 4.973	
y = sigmoid (2)	the tot set sawary 211
ritura (y)	point (+ sauracy on the test set; (account 21)
0	mateix (y-text
	of metaus Confusion of
	Confusion matrix = retries confusion matrix (g. tri
V VIII	( Triplands)

	4	Date
cm-display plot ()	-	Now.
cm-display plot () plt. show()	ciin	There were no mining or incomutert value
0/0 - 0.83 6360 - 4/0	diir	The Confusion matrix should achieved 25.240 accuracy.
unray on the train it: 1.00	civ	accuracy.
have where twice	Cix	Reptiles and amphibians was mostly mislaufied dans. They share derived a large natures.
MR - Cemmia - sep. (SV year durate) months and		They share removed features.  Moramals and high also as they had our byroing features.
stujution level: lower estigation - higher turnous		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
ingen tomore .	(	Me 21 1 - 21 1 - 21 1 1 1 1 1 1 1 1 1 1 1
in spent in company: longer tenure -> higher libited		P105.0 =
ology: Lew solvey -> higher attriction postburit: Sala 4 tubmid now higher turnown		Ju 00 11
partment: Sala & technical have higher turniones		2
X = in deep ( spice side ( shows ))		2 20 - (21) 100/ 41- 3 [-8 41] Male
ty 35.240 Privated = 1		2001 100 4 2 - (11) 100 11 10 100 10 100 10 10 10 10 10 10
it is a good accuracy, but accuracy alone con		2 (10 2 ) bijsa)
Muleading due to potential class imbalance.	-	12 150 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
ithan wood ithur comment it in - wat		10000
o datail		0 = ( + 10) ( 10)
o datail this has been been to	-	03
, penformed data processing steps.		O - L-B will hamil
and the animal same column standard by lead	in the second	p1950 = (ap 3) who
, conformed data professing stope.  And the animal same column, standardized the failing it damaded given to improve mostly performance.	1	P181-3 - 0 - 0 - 101-0 - ( 14 - 2 ) 1/18
	1	

```
Code
import pandas as pd
import numpy as np
df=pd.read_csv("/content/HR_comma_sep.csv")
df.head(3)
print(df.isnull().sum())
print(df.groupby('left').mean(numeric_only=True))
print(df.groupby('salary').mean(numeric_only=True))
import matplotlib.pyplot as plt
pd.crosstab(df.salary,df.left).plot(kind='bar')
plt.title('Employee Retention vs Salary')
plt.xlabel('Salary')
plt.ylabel('Number of Employees')
plt.show()
pd.crosstab(df.Department,df.left).plot(kind='bar')
plt.title('Employee Retention vs Department')
plt.xlabel('Department')
plt.ylabel('Number of Employees')
plt.show()
salary_dummies = pd.get_dummies(df.salary, prefix="salary")
dept_dummies = pd.get_dummies(df.Department, prefix="dept")
```

```
df_with_dummies = pd.concat([df, salary_dummies, dept_dummies], axis=1)
df_with_dummies = df_with_dummies.drop(['salary', 'Department'], axis=1)
X_features = ['satisfaction_level', 'last_evaluation', 'number_project', 'average_montly_hours',
'time spend company', 'Work accident', 'promotion last 5years'] + list(salary dummies.columns) +
list(dept_dummies.columns)
X = df\_with\_dummies[X\_features]
y = df_with_dummies.left
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
from sklearn.metrics import accuracy_score
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy of the model:", accuracy)
```

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



Decision Tree		
import pandae as pd.		
from skleam tree import pairier factorifer		0 3 0 0 0
from Skleam metric import train tell soft	Atual	00500
import reaborn a cru		000110
impart matplotlib pyplot a plt		1000015
df = pd. read csv ( 1/cont. of 1/2		the Redicted to the same of th
df = pd . Trad csv ('/ Corlini / drug .csv')  X = pd .gtduromies (df. drop ("Porug"), aix =1 ?)  4 = dl ("Drun"?		Daniella San Allena Care
y = of [ Drug ]	-	the interior distriction to
h = of cound, ]		0/6 -
X - Main, X - Teil, X - Train, X, test = train_test_split(x, y,	- In	Deam absolute erron: 571-61 sein mut Te hy
tut_cize = 0.2 )		san Equaned error: 6338.2
dtree = Decison Tree (lacifier () dtree = jit (x = train, y = train)	R	not mean equated error: 79.6128
u oned = dtree onedit (x-text)	1 x ) + 1 1	vertically vertically that I have been and
y pred = dtree, predict (x-ted)	7 . A.	A - 900 Lift
accuracy = accuracy - More (y-ted, y-prod)	3-1- 1	wray was 1.0 for ins detact
aumay = aumay - More (4 ta, y - pres) prior (4 " Brunay : Saumay ") cm = confuion matrix (4 ta, 4 - pred) plk igure ( jogice = (8, 1))	Th	2 Confision matrix only has diagonal them +
cm = confusion matrix (y text, y-pred)	mec	le Confusion matrix only has diagonal element;
plt. sigure (sigsize = (8,6))		Charles Francis Common
ptt. show ()		(1) I want to him with the same
	!- Key	Seatures like population density and awage in
0/8 —	ang	Seature like oppulation domity and awage in
		Con to
Aumay: 1.0		
10.0 1		- 1/0
		Armay 1.0

```
Code
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn import tree
import matplotlib.pyplot as plt
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)
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
print("Confusion Matrix:\n", conf_matrix)
plt.figure(figsize=(12, 8))
tree.plot_tree(clf, feature_names=iris.feature_names, class_names=iris.target_names, filled=True)
plt.show()
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn import tree
import matplotlib.pyplot as plt
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)
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
```

```
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
print("Accuracy:", accuracy)
print("Confusion Matrix:\n", conf_matrix)
plt.figure(figsize=(12, 8))
tree.plot_tree(clf, feature_names=iris.feature_names, class_names=iris.target_names, filled=True)
plt.show()
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np # import numpy
data = pd.read_csv("petrol_consumption.csv")
X = data[['Petrol_tax', 'Average_income', 'Paved_Highways',
      'Population_Driver_licence(%)']]
y = data['Petrol_Consumption']
```

```
X_train, X_test, y_train, y_test = train_test_split(
  X, y, test_size=0.2, random_state=42)
regressor = DecisionTreeRegressor()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
plt.figure(figsize=(15, 10))
# Assuming 'data' is your original pandas DataFrame
plot_tree(regressor, feature_names=data[['Petrol_tax', 'Average_income', 'Paved_Highways',
'Population_Driver_licence(%)']].columns, filled=True, rounded=True)
plt.show()
```

### PROGRAM 6 Build KNN Classification model for a given dataset.

Pearson	Age	Salary 50	Target	Distance	Lab-6 topp min fund
B	23	55	N	52.8	Wirthert Onnday as all
C	24	70	N	31.95	impart realearn as sins from skylam model white
0	41	60	У	40.44	from sklepen med de sins
F	43	70	y	31-04	frem sklearn model relation imposit train that split
F	38	40	у	80.09	from Schann model neighbours import Knuighbours (kaifun
X	35	100	?		inis data - ad and com
					v inis = pd. read sv ( inis data .csv )
Testingo .	Calle	- Makes			X inix = irix data ilor [:, -1]
Distance :	= \ (7.	-x)2+ (yo-4	1 12+		X one = true data lor [:, -1] pure harmon
Quitani.	VIUO	300	0		X-train_inis, X-test_inis, Y-test_inis (x-inis, Y-inis,
800 K = 2	Dani	in avende	na anden		tett_size = 0.2 , nandem state
Jon K-	Juliano	e an oncens	9	429	k = 3
Distance	Doub	Tanat			- knn_claufer = KNughtours (n_nughtours = k)
31.04	Nun	Tanget		F172000	Khri-classifin sit (x Inain in , y Inain in)
31.95	2	v		1400000	a course sig half a super whiten and ston pant of
40.44	3	v /			accuracy = accuracy - were (y tet iris, y prid - iris)
10.17	,	*			Conf matrix = Conf matrix (y tut, y pria min)
c -	. + .	1 1 1	) F	- 4 11	Conf matrix = leng matrix (y tid , y pad sii)
since ma	writy is	Yes, Jon C:	15,1000 10	vigu will	227
yes.	/	1		7000	print (f " Augracy Score: Sacuracy ?")
-	0	/		1019	print (fi human score ; acumany ; print (clau mean) print (clau mean)
	T			100	orient ( clay report)
BLARE	/				- Plura G Charles
				1	TO THE RESERVE TO THE
Control	MAN A	I m co			1. (1) 6561 9 (1)
					- OIP
Carpella		1215 mm	DOT BE		Of P Accuracy More: 1.0
THE REAL PROPERTY.					Gangarian: 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
	4 4				0011
	The same of				

		Date
100		Date Page
Classification	m Report	100-6
	Pruision	Recall fi-une supe
Setosa	1.00	1.00 1.00
vericolor	1.00	1.00 1.00 9
	1.00	1.00 1.00 11
<i>winginica</i>	Mary June Contract	
acuracy	100 I	1.00 30
mairo aug	1.00 .	
wiighted aug	Danie tot Y	V. team this v stat fine.
them of a	In the state of the state of	63.6
MINISTER CONTRACTOR		1010/2
11 0 0:	Linnela no	have but amenally
		THE PLAN SECTION AND A SECTION
K = 3, yu	an nata = 1-	unacy home but generally
K = 3, yar	es rate = 1-	according = 0.0 (no miscle
to find by	at k, ted mu	thiple value & plot the inner
to find by	at k, ted mu	thiple value & plot the inner
k=5, and to	at k, ted mu	acidicy = 0.0 ( no miche
K=5, will to find be	at k, teil mu	diple value 4 plot the ener
K=5, will to find be	at k, teil mu	diple value 4 plot the ener
K=5, will to find be	at k, teil mu	diple value 4 plot the ener
K=5, will to find be	at k, teil mu	diple value 4 plot the ener
K=5, will to find be	at k, teil mu	acidicy = 0.0 ( no miche
K=5, will to find be	at k, teil mu	diple value 4 plot the ener
K=5, will to find be	at k, teil mu	diple value 4 plot the ener
K=5, will to find be	at k, teil mu	diple value 4 plot the ener
K=5, will to find be	at k, teil mu	acitiscu = 0.0 (no muche utiple Salve + plot the more calina nave diff range ( sond galane valer inque nordizing data . Imprevis = 70.07 % after galling
K=5, will to find be	at k, teil mu	diple value 4 plot the ener
K=5, will to find be	at k, teil mu	acitiscu = 0.0 (no muche utiple Salve + plot the more calina nave diff range ( sond galane valer inque nordizing data . Imprevis = 70.07 % after galling

```
Code
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt
try:
  data = pd.read_csv('/content/iris (1).csv')
except FileNotFoundError:
  print("Error: 'iris.csv' not found. Please upload the file to your Colab environment.")
  exit()
X = data.drop('species', axis=1)
y = data['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Accuracy Score:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:")
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
       xticklabels=knn.classes_, yticklabels=knn.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import matplotlib.pyplot as plt
try:
  diabetes = pd.read_csv('diabetes.csv')
except FileNotFoundError:
  print("Error: 'diabetes.csv' not found. Please ensure the file is in the current directory.")
```

```
X = diabetes.drop('Outcome', axis=1)
y = diabetes['Outcome']
scaler = StandardScaler()
X = scaler.fit\_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

exit()

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
try:
  heart = pd.read_csv('heart.csv')
except FileNotFoundError:
  print("Error: 'heart.csv' not found. Please ensure the file is in the current directory.")
  exit()
X = heart.drop('target', axis=1)
y = heart['target']
scaler = StandardScaler()
X = scaler.fit\_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
best_k = 1
best_accuracy = 0
```

```
for k in range(1, 21):
  knn = KNeighborsClassifier(n_neighbors=k)
  knn.fit(X_train, y_train)
  y_pred = knn.predict(X_test)
  accuracy = accuracy_score(y_test, y_pred)
  if accuracy > best_accuracy:
     best_accuracy = accuracy
     best_k = k
print(f"Best k: {best_k} with accuracy {best_accuracy}")
knn = KNeighborsClassifier(n_neighbors=best_k)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
```

```
plt.show()
print("Classification Report:")
print(classification_report(y_test, y_pred))
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report, confusion_matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
print(classification_report(y_test, y_pred))
# prompt: For Iris dataset
# How to choose the k value? Demonstrate using accuracy rate and error
# rate. Give theory
import pandas as pd
from sklearn.model_selection import train_test_split
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
# Load the Iris dataset
try:
  data = pd.read_csv('/content/iris (1).csv')
except FileNotFoundError:
  print("Error: 'iris (1).csv' not found. Please upload the file to your Colab environment.")
  exit()
# Prepare the data
X = data.drop('species', axis=1)
y = data['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Scale the data (important for KNN)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_{\text{test}} = \text{scaler.transform}(X_{\text{test}})
# Find the optimal k value
```

```
error_rates = []
for k in range(1, 31): # Test k values from 1 to 30
  knn = KNeighborsClassifier(n_neighbors=k)
  knn.fit(X_train, y_train)
  y_pred = knn.predict(X_test)
  error_rates.append(1 - accuracy_score(y_test, y_pred)) # Error rate = 1 - accuracy
# Plot error rates
plt.figure(figsize=(10, 6))
plt.plot(range(1, 31), error_rates, color='blue', linestyle='dashed', marker='o',
     markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
plt.show()
# Theory for choosing k:
# The optimal 'k' value minimizes the error rate.
# Very small k (e.g., 1) can lead to overfitting, being too sensitive to noise.
# Very large k (e.g., 30) can lead to underfitting, smoothing out the decision boundaries too much.
# We seek a k that balances these extremes, as shown by the error rate plot.
#Select k based on the minimum error rate observed in the plot
best_k = error_rates.index(min(error_rates)) + 1 #Add 1 as the index starts from 0
# Train and evaluate the model with the best k
```

```
knn = KNeighborsClassifier(n_neighbors=best_k)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
# Evaluate the model
print("Accuracy Score:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:")
cm = confusion_matrix(y_test, y_pred)
print(cm)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
       xticklabels=knn.classes_, yticklabels=knn.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

## import matplotlib.pyplot as plt

# Plot

```
# Load data
df = pd.read_csv('/content/iris (1).csv')
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
# Store accuracy and error rate
accuracy = []
error_rate = []
# Try k from 1 to 20
for k in range(1, 21):
  knn = KNeighborsClassifier(n_neighbors=k)
  knn.fit(X_train, y_train)
  preds = knn.predict(X_test)
  acc = accuracy_score(y_test, preds)
  accuracy.append(acc)
  error_rate.append(1 - acc)
```

```
plt.figure(figsize=(10,5))
plt.plot(range(1, 21), accuracy, label='Accuracy')
plt.plot(range(1, 21), error_rate, label='Error Rate')
plt.xlabel('K Value')
plt.ylabel('Rate')
plt.title('K vs Accuracy and Error Rate')
plt.legend()
plt.show()
import pandas as pd
from sklearn.preprocessing import StandardScaler
# Load data
df = pd.read_csv('/content/diabetes.csv')
X = df.drop('Outcome', axis=1) # Features
y = df['Outcome']
                          # Target
# Perform scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Convert back to DataFrame (optional)
X_scaled_df = pd.DataFrame(X_scaled, columns=X.columns)
```

## PROGRAM 7 Build Support vector machine model for a given dataset

# Screenshot

Dots	bas
P(8) = 10   10   10   10   10   10   10   10	Out e
to clavily the following going times sym	$\alpha_1(6) + \alpha_2(9) + \alpha_3(9) = +100   \alpha_1 = 13/4$ $\alpha_1(9) + \alpha_2(6) + \alpha_3(9) = +1   \alpha_2 = 13/4$
(1,1)(2,1)(1,-1)(2,-1)(4,0)(5,1)(5,-1)(6,0)	$\omega = \alpha_1 \widetilde{S_1} + \alpha_2 \widetilde{S_2} + \alpha_3 \widetilde{S_3}$
(1)(2,1)(1,-1)(2,-1) } +ve	W = x15, + x252 + x253
(9,0)(s,1)(s,-1)(6,0) 3 -ve	$= \frac{13}{4} \left\{ \begin{array}{c} 2 \\ 1 \\ 1 \end{array} \right\} + \frac{13}{4} \left\{ \begin{array}{c} 2 \\ 1 \\ 1 \end{array} \right\} + \frac{2}{4} \left\{ \begin{array}{c} 2 \\ 1 \\ 1 \end{array} \right\} + \frac{2}{4} \left\{ \begin{array}{c} 4 \\ 1 \\ 1 \end{array} \right\} + \frac{2}{4} \left\{ \begin{array}{c} 4 \\ 1 \\ 1 \end{array} \right\} = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}$
A Train of Table of Table of the Table of the Table of the Table of Table of the Ta	= - [1]
\$ 5 = 6   14   = 9	[-3]
in more and continued and a second	$2 = \begin{bmatrix} +1 \\ 0 \end{bmatrix}  b = -3$ $b+3 = 0$
-5 -4 -3 -1 0 4 5 6	intercept on x-axis = 3 2(1) -> line parallel to y-ais
2 3	
1	\$ 0,0
$S_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}  S_2 = \begin{bmatrix} 2 \\ -1 \end{bmatrix}  S_3 = \begin{bmatrix} 4 \\ 0 \end{bmatrix}$	the secretary and the second second
$S_{1}^{-} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}  S_{2}^{-} = \begin{bmatrix} 4 \\ 1 \end{bmatrix} $	[6] - 2 [2] - 2 [6] -
1+= 33 2 1 2 2	[ ] = 45 [ ] = 85 [ ] = 4
	[0] = 3 [1] [0] [0] 3
4/2/25 + 402523 + 43 2325 = -1	1+ = 33, x + 3
	1 = 2 2 2 2 1 = 2 2 2 2 1 + 2 2 2 2 2 2 2 2 2 2 2 2 2 2

_	
DOM:	limen sum aumany: 1.0 - mus de
Code SVM -	Confusion matrix
germ sklearn model whating	(10 0 0) AVE traying man more man
imposit pandar as pd from skkann medel - sleitum imposit train not split from sklavn surn imposit svc from sklavn mitnis imposit acumay-casis, lenguin matrix	0 9 0 11 varian
inix dl = pd. read con C " "	Quitonit = 1 = 20 miles 2 and 1 xin = x
Y = iris de ["apeier"]	Quitoni = = = = = = = = = = = = = = = = = = =
(x, y, ted size = 0.2, nandom state = 12)	Both RBF and trinear gour the varie according was
svm_nbf = svc(kund = 'nbf', nandem_iddi = +2 svm_inean.fit(x_tnain, y_tnain) y_prid_linean = svm_linean, pridit(x_tid)	The AUC More was 35 - 38 %. It shows that the
y pred linear = wm linear predict (x ted)	AUC model performed pretty well for the bellio -
print C"RBF Aumary":, aumary-rure (y tod, )	The AUC MARK for low inch cost was 1.00 as it we
print ( "In livien away ", away wore (y. I.d., y.	The AUC work for for inc. CSV was 1.00 and we way small and chan dataset with wall expendite claw
The said of the sa	9/2
OP - RBF SVM ammany: 1.0	REE SUM annound 11.0
Confusion matrix -	(infinism materix
0 3 0	000:
0 9 0	0 8 0

```
Code
import numpy as np
import matplotlib.pyplot as plt
positive_class = np.array([[4, 1], [4, -1], [6, 0]])
negative_class = np.array([[1, 0], [0, 1], [0, -1]])
plt.figure(figsize=(8, 6))
plt.scatter(positive_class[:, 0], positive_class[:, 1], color='red', label='Positive Class', s=100,
edgecolors='black')
plt.scatter(negative_class[:, 0], negative_class[:, 1], color='blue', label='Negative Class', s=100,
edgecolors='black')
all_points = np.concatenate([positive_class, negative_class])
labels = ["(4,1)", "(4,-1)", "(6,0)", "(1,0)", "(0,1)", "(0,-1)"]
for i, txt in enumerate(labels):
  plt.annotate(txt, (all_points[i][0], all_points[i][1]), textcoords="offset points", xytext=(0,5),
ha='center', fontsize=10)
x_values = np.linspace(-1, 7, 100)
y_values = np.zeros_like(x_values)
plt.plot(x_values, y_values, color='black', linestyle='--', label='Optimal Hyperplane (y = 0)')
```

```
plt.plot(x_values, y_values + 1, color='gray', linestyle=':', label='Margin at y = 1')
plt.plot(x_values, y_values - 1, color='gray', linestyle=':', label='Margin at y = -1')
plt.title('Optimal Hyperplane for SVM (Visual Approximation)', fontsize=14)
plt.xlabel('x1')
plt.ylabel('x2')
plt.xlim(-1, 7)
plt.ylim(-2, 2)
plt.axhline(0, color='black',linewidth=0.5)
plt.axvline(0, color='black',linewidth=0.5)
plt.legend()
plt.grid(True)
plt.show()
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
data = pd.read_csv('/content/iris (1) (1).csv')
X = data.drop('species', axis=1)
```

```
y = data['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
svm_rbf = SVC(kernel='rbf')
svm_rbf.fit(X_train, y_train)
y_pred_rbf = svm_rbf.predict(X_test)
accuracy_rbf = accuracy_score(y_test, y_pred_rbf)
cm_rbf = confusion_matrix(y_test, y_pred_rbf)
print("SVM with RBF Kernel:")
print("Accuracy:", accuracy_rbf)
print("Confusion Matrix:\n", cm_rbf)
plt.figure(figsize=(6, 4))
sns.heatmap(cm_rbf, annot=True, fmt='d', cmap='Blues',
       xticklabels=data['species'].unique(),
       yticklabels=data['species'].unique())
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix (RBF Kernel)')
plt.show()
svm_linear = SVC(kernel='linear')
svm_linear.fit(X_train, y_train)
```

```
y_pred_linear = svm_linear.predict(X_test)
accuracy_linear = accuracy_score(y_test, y_pred_linear)
cm_linear = confusion_matrix(y_test, y_pred_linear)
print("\nSVM with Linear Kernel:")
print("Accuracy:", accuracy_linear)
print("Confusion Matrix:\n", cm_linear)
plt.figure(figsize=(6, 4))
sns.heatmap(cm_linear, annot=True, fmt='d', cmap='Blues',
       xticklabels=data['species'].unique(),
       yticklabels=data['species'].unique())
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix (Linear Kernel)')
plt.show()
import numpy as np
import matplotlib.pyplot as plt
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_curve, auc
import seaborn as sns
from sklearn.preprocessing import label_binarize
```

```
from sklearn.multiclass import OneVsRestClassifier
data = pd.read_csv('/content/letter-recognition.csv') # Replace with the correct path if necessary
X = data.drop('letter', axis=1)
y = data['letter']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
svm_classifier = SVC(kernel='rbf', probability=True) # probability=True is needed for ROC curve
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
print("SVM Classifier:")
print("Accuracy:", accuracy)
print("Confusion Matrix:\n", cm)
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=np.unique(y),
yticklabels=np.unique(y))
```

plt.xlabel('Predicted')

```
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
y_test_bin = label_binarize(y_test, classes=np.unique(y))
n_{classes} = y_{test_bin.shape[1]}
classifier = OneVsRestClassifier(SVC(kernel='rbf', probability=True))
classifier.fit(X_train, y_train)
y_score = classifier.predict_proba(X_test)
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
  fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_score[:, i])
  roc_auc[i] = auc(fpr[i], tpr[i])
fpr["micro"], tpr["micro"], _ = roc_curve(y_test_bin.ravel(), y_score.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
plt.figure(figsize=(8, 6))
plt.plot(fpr["micro"], tpr["micro"],
     label='micro-average ROC curve (area = {0:0.2f})'
         ".format(roc_auc["micro"]))
```

```
plt.plot([0, 1], [0, 1], 'k--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Micro-averaged ROC Curve')

plt.legend(loc="lower right")

plt.show()

print(f"Micro-averaged AUC: {roc_auc['micro']}")
```

PROGRAM 8 Implement Random forest ensemble method on a given dataset.

## Screenshot

		Dote	Done
	lab=8	- Algertha of molow	-> Algorithm of random Forest
	Difference between Decision	ers tree and Randons Facet	1) Topat dotant O with features of days labels 2) Sot the no of trees n etionation 3) for each tree is from 1 to n etumature
-	Model type is single	Model type is ememble of	mely tipe a rough though tipe a mouth
- 100	Prone to overstitling	Less prone due to ourraging	Code: impart pandar as pd
	complex data	accuracy higher due to ensemble learning	Jaem sklear medd slutter impeat train tid split Jaem sklearn gaemble impeat Randomferet Claufer Jaem sklearn melau impeat auwaeg wege
+>	Facter to train and	Slower due to muttiple trees	vivis = pot read esv ( ini csv ) Dont at interest
<b>E</b> >	low bias, high vaniance	low bias, reduced variance	they bird, high unaitages law hear, and was to
→ · · · · · · · · · · · · · · · · · · ·	Panameters of RF	Tall	oc train, oc teil, y train, y teil, = train. tet. (21/2). Led, size = 0.2, random - date = 42
6	n_eitimatorx No. 6 Clax_wiight — weight max_depth — max_dep bootstrap — whether bes	pth of each tree other amples are used when	of refault = Randomport Claufus ( h_utanter 10, pardom = stat = 12)  If default of (x-train, y train)  J-prud-default 0 = 11, defaultprudit (x tat)
	n jobs - na of jobs nandom state - med	to run in parallel	away difault = accuracy were (y text, y god)  print (away signit : 4)  when there is no so, 100, 200, 500
	ventose - Control the	unbosity of output	Ent (work)  part (but n)

	Page_
ole - awray	with $n = etimatan = 10:1.$
J,	= 90:
the seal of	2 100
I want by Fine	500 :
But n- externator	:10
but-aurany-	were : 1
0/	
	7
	the state of the state of the
and the second	the her model
	county drawn to be a second
Comment than to	
	to a state of the same of the
	the appealant letter
	9
	- Down His
All was to sell the	mit Marcal sure party
	turn de
	A The March and there of
- 1 3 Au	and always at
de la lace	-t. ida
	The state of the s

```
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
# Load the dataset
df = pd.read_csv('/content/iris (1).csv')
# Prepare features and target
X = df.drop(columns=['species']) # Assuming 'species' is the target column
y = df['species']
# 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)
# Build Random Forest with default n_estimators (10)
rf_default = RandomForestClassifier(n_estimators=10, random_state=42)
rf_default.fit(X_train, y_train)
y_pred_default = rf_default.predict(X_test)
# Measure accuracy
default_score = accuracy_score(y_test, y_pred_default)
```

```
print(f"Default RF accuracy (n_estimators=10): {default_score:.4f}")
# Fine-tune the number of trees
scores = []
n_range = range(1, 101)
for n in n_range:
  rf = RandomForestClassifier(n_estimators=n, random_state=42)
  rf.fit(X_train, y_train)
  y_pred = rf.predict(X_test)
  score = accuracy_score(y_test, y_pred)
  scores.append(score)
# Find the best score and number of trees
best\_score = max(scores)
best_n = n_range[scores.index(best_score)]
print(f"Best RF accuracy: {best_score:.4f} with n_estimators={best_n}")
# Optional: Plot accuracy vs number of estimators
plt.figure(figsize=(10, 6))
plt.plot(n_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()
```

## PROGRAM 9 Implement Boosting ensemble method on a given dataset.

#### Screenshot

-14	Date Page
•	
. 178	100-9
	11.82
	Boosting: Combines multiple weak learners to south
	a during learner . It works by training
	model aguertially where each model to
	model significally where each model for on wrong model by premous one
	Parameters:
	estimater The base model
	a it to so of week learner
	learning note shrinks contribution of each learner
	algarithm 'SAMME-K'
	Random-nate for suproducibility
	Algo:
1	Start with equal sots for all training sample
2.	Train a weak model
3	Cal sonor & sendate somale with
t.	Cal war 4 update sample with a with a with and an
	QUIDAUL.
	auray.
,	Evil and tim
,	Final prediction

#### Code

import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model\_selection import train\_test\_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy\_score
from sklearn.tree import DecisionTreeClassifier

```
# Load dataset
df = pd.read_csv("/content/income.csv")
# Drop rows with missing values
df.dropna(inplace=True)
# Encode categorical columns
label_encoders = {}
for column in df.select_dtypes(include=['object']).columns:
  le = LabelEncoder()
  df[column] = le.fit_transform(df[column])
  label_encoders[column] = le
# Separate features and target
X = df.drop(columns=['income_level'], errors='ignore', axis=1)
y = df['income_level']
# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# AdaBoost with 10 estimators
model_10 = AdaBoostClassifier(n_estimators=10, random_state=42)
model_10.fit(X_train, y_train)
y_pred_10 = model_10.predict(X_test)
score_10 = accuracy_score(y_test, y_pred_10)
print(f"Accuracy with 10 estimators: {score_10:.4f}")
# Fine-tune number of estimators
best\_score = 0
```

```
best_n = 0
estimators_range = list(range(10, 201, 10))
scores = []
for n in estimators_range:
  model = AdaBoostClassifier(n_estimators=n, random_state=42)
  model.fit(X_train, y_train)
  y_pred = model.predict(X_test)
  score = accuracy_score(y_test, y_pred)
  scores.append(score)
  print(f"n_estimators={n}, Accuracy={score:.4f}")
  if score > best_score:
     best_score = score
     best_n = n
print(f"\nBest Accuracy: {best_score:.4f} using {best_n} estimators")
# Plot accuracy vs number of estimators
plt.figure(figsize=(7, 4))
plt.plot(estimators_range, scores, marker='o', linestyle='-', color='blue')
plt.title("Accuracy vs Number of Estimators (AdaBoost)")
plt.xlabel("Number of Estimators (Trees)")
plt.ylabel("Accuracy")
plt.grid(True)
plt.xticks(estimators_range)
plt.tight_layout()
plt.show()
```

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

#### Screenshot

	100-10
	K-Means
	Partition data into k clusters by:
1.	
2.	Assigning points to she nearest gentloid
3.	updating rentroids based on assigned points
4.	Randomly initializing untroids  Assigning points to the nearest controid  updating rentroids based on assigned points  Repeating until someorgense
10	
->	Choosing number of clusters
	Elbow Method
	Silhouette Score
•	Elbow Method Silhouette Score Domain Knowledge
->	Sum of squared emars
	Sum of squared errors $SSE = i = 15 kx E Ci \Sigma   x - ui  ^2$
	Key Panameter
	n cluster No of electers to from
	initialization method
THE WAR	n init No. of initialization
	max iteration per run
	Control randomner
	Convergence Mentiona
	algorithm to un

#### Code

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

from sklearn.metrics import accuracy\_score

```
from scipy.stats import mode
import matplotlib.pyplot as plt
# Step 1: Generate sample data and save to CSV
np.random.seed(42)
names = [f"Person_{i}]" for i in range(50)]
ages = np.random.randint(20, 60, 50)
income = np.random.randint(30000, 120000, 50)
df = pd.DataFrame({'Name': names, 'Age': ages, 'Income': income})
df.to_csv("income.csv", index=False)
# Step 2: Load the data
data = pd.read_csv("income.csv")
# Drop 'Name' and extract features
X = data[['Age', 'Income']]
# Step 3: Split the data
X_train, X_test = train_test_split(X, test_size=0.2, random_state=42)
# Step 4: Perform scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
```

```
X_test_scaled = scaler.transform(X_test)
# Step 5: Plot SSE vs number of clusters (Elbow method)
sse = []
k_range = range(1, 11)
for k in k_range:
  kmeans = KMeans(n_clusters=k, random_state=42)
  kmeans.fit(X_train_scaled)
  sse.append(kmeans.inertia_)
plt.figure(figsize=(8, 4))
plt.plot(k_range, sse, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('SSE (Inertia)')
plt.title('Elbow Method For Optimal k')
plt.grid(True)
plt.show()
# Step 6: Choose optimal number of clusters (say 3) and fit model
optimal_k = 3
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
kmeans.fit(X_train_scaled)
# Predict on test data
```

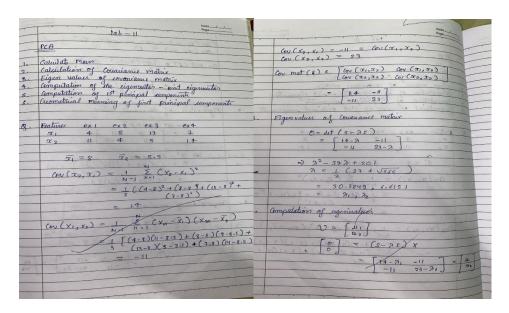
```
predictions = kmeans.predict(X_test_scaled)
# Note: There's no ground truth labels, but for demonstration,
# we can try assigning true clusters (via KMeans on full data)
# and see if predicted clusters align
# Fit on full data to assign pseudo-labels
full_kmeans = KMeans(n_clusters=optimal_k, random_state=42)
true_clusters = full_kmeans.fit_predict(scaler.fit_transform(X))
# Align predicted clusters using majority voting (only for demonstration)
# Match predicted labels to closest true labels
def map_clusters(true_labels, pred_labels):
  labels = np.zeros_like(pred_labels)
  for i in range(optimal_k):
    mask = (pred\_labels == i)
    if np.sum(mask) == 0:
       continue
    labels[mask] = mode(true_labels[mask])[0]
  return labels
mapped_preds = map_clusters(true_clusters[X_test.index], predictions)
accuracy = accuracy_score(true_clusters[X_test.index], mapped_preds)
print(f"Approximate Clustering Accuracy: {accuracy:.2f}")
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
# Step 1: Load Iris dataset
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['target'] = iris.target
# Keep only petal length and petal width
X = df[['petal length (cm)', 'petal width (cm)']].values
# Step 2: Check impact of scaling
# Try without scaling
sse_unscaled = []
for k in range(1, 11):
  kmeans = KMeans(n_clusters=k, random_state=42)
  kmeans.fit(X)
  sse_unscaled.append(kmeans.inertia_)
```

```
# Now scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
sse_scaled = []
for k in range(1, 11):
  kmeans = KMeans(n_clusters=k, random_state=42)
  kmeans.fit(X_scaled)
  sse_scaled.append(kmeans.inertia_)
# Step 3: Plot Elbow Comparison (Scaled vs Unscaled)
plt.figure(figsize=(10, 5))
plt.plot(range(1, 11), sse_unscaled, marker='o', label='Unscaled')
plt.plot(range(1, 11), sse_scaled, marker='s', label='Scaled')
plt.title('Elbow Method (Petal Features Only)')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('SSE (Inertia)')
plt.legend()
plt.grid(True)
plt.show()
```

PROGRAM 11 Implement Dimensionality reduction using Principal Component Analysis (PCA) method.

#### Screenshot



				Di Po	nte
3	$\frac{u_1}{n} = \frac{1}{n}$	12 -	DC		9
	u, = 113	L, 112	= (14-2,	) <del>+</del> ,	
	u,	- [11 %	]		
To	fond eigen	vertor, me	compete	length o	g ×1
	1/4/11	= 112	+(4-2,)2		
		= 19.			
	- unit	eigenwecter panding t	21 01	= [1/1	2,)/II.M.II.
C2	0.89	94			
Comp	putation of	1st prins	ipal com	ponenti	
	DC2H X2H				
e, <sup>T</sup>	X1K - X, X2K - X2		.5574 - 1		XIR - XI YEIG - XZ
		= 0	.55 74 (XI	x -7,)-	0.8303 (x2
		A STATE OF THE PARTY OF THE PAR	-		

```
Code
import pandas as pd
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
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
#1. Load data
df = pd.read_csv("heart.csv")
# 2. Label-encode binary text columns
le = LabelEncoder()
for col in ["Sex", "ExerciseAngina"]:
  df[col] = le.fit_transform(df[col])
```

df[col] = le.fit\_transform(df[col])
# 3. Separate features and target
X = df.drop("HeartDisease", axis=1)
y = df["HeartDisease"]

```
# 4. Build preprocessing pipeline:
  - One-hot for multi-category columns (using sparse output=False)
   - passthrough the rest
   - then scale everything
cat_cols = ["ChestPainType", "RestingECG", "ST_Slope"]
preprocessor = Pipeline([
  ("onehot", ColumnTransformer([
     ("ohe", OneHotEncoder(sparse_output=False, drop="first"), cat_cols)
  ], remainder="passthrough")),
  ("scaler", StandardScaler())
1)
# 5. Apply preprocessing
X_proc = preprocessor.fit_transform(X)
# 6. Train/test split
X_train, X_test, y_train, y_test = train_test_split(
  X_proc, y, test_size=0.2, random_state=42
)
#7. Define models
models = {
  "SVM": SVC(random_state=42),
  "LogisticRegression": LogisticRegression(max_iter=1000, random_state=42),
```

```
"RandomForest": RandomForestClassifier(random_state=42)
}
# 8. Train & evaluate before PCA
print("=== Accuracies BEFORE PCA ===")
scores_before = {}
for name, clf in models.items():
  clf.fit(X_train, y_train)
  preds = clf.predict(X_test)
  acc = accuracy_score(y_test, preds)
  scores_before[name] = acc
  print(f"{name:17s}: {acc:.4f}")
# 9. Apply PCA (retain 95% variance)
pca = PCA(n_components=0.95, random_state=42)
X_train_pca = pca.fit_transform(X_train)
X_{test_pca} = pca.transform(X_{test})
print(f"\nPCA retained {pca.n_components_} components, "
   f"explained variance = {pca.explained_variance_ratio_.sum():.4f}\n")
# 10. Train & evaluate after PCA
print("=== Accuracies AFTER PCA ===")
scores_after = {}
for name, clf in models.items():
  clf.fit(X_train_pca, y_train)
```

```
preds = clf.predict(X_test_pca)
acc = accuracy_score(y_test, preds)
scores_after[name] = acc
print(f"{name:17s}: {acc:.4f}")
```