VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.



LAB REPORT on

Machine Learning (23CS6PCMAL)

Submitted by

Aakanksha V R(1BM22CS001)

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 **Aakanksha V R (1BM22CS001),** who is bonafide student of **B.M.S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements in respect of an Machine Learning (23CS6PCMAL) work prescribed for the said degree.

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

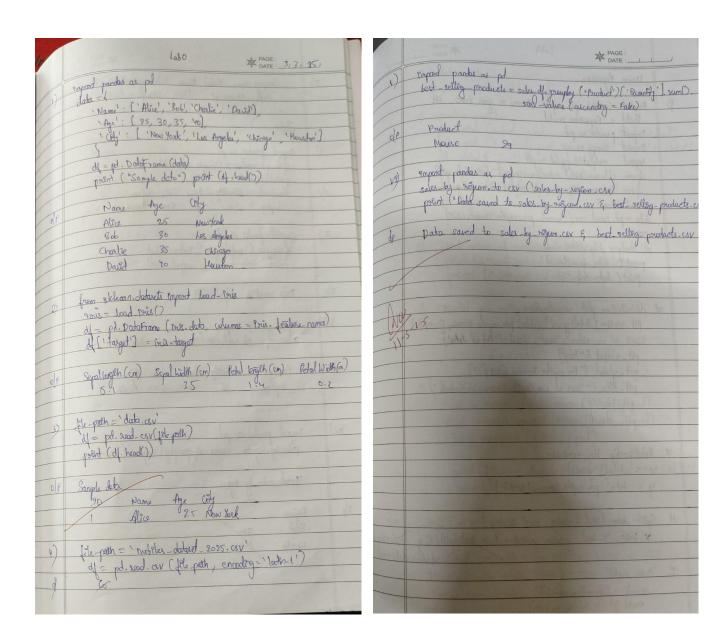
Index

Sl. No.	Date	Experiment Title	Page No.	
1	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				

Program 1

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

	PAGE:
	LI HARE WHAT KAM C
ole	Company Name Needel Name Hobbe Night KAM Frank Com Apple (Phosis NZ 19868 1749 668 12Mp
	Soles
1)	topost parties as pol - pol grad csv ('comple sales data csv')
olb	toppad pards as pel 4 - pr. and csv ('sample sales data cev') prent (d. bod(')) Pacduct Quantity Price Sales Region Lapter 5 1000 5000 Worth
97)	empost parks as pd dite cer ('output csv', 'Index = Falu) pront ('Doto saved to output car')
ole	poto savat so output. av
(ti)	ingust parely as pd nies of - pt. mod-csv ('sample sales dala.cev')
olp	Product auntity Price Sales legten
	for delicating of
iii)	Impart pandae as pol soles by segren sales of grouply (Region) (Sales I sund)
	Total sales by leien
2/4	Region Hard
	North 16400
	Total John State of the State o



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

	Lob1 AL PACE:
	PAGE: 13 125
	Howard Demonstrate vossous data propriating kellingers for
	Housing Demonstrate various data propriating belongers for
	0
	Ingrad produk as god
1	sales d = pd. sind-csv ('handig - cav')
	poset ("Sample dates")
12	port (sola-di-had())
	11.1 (4)
#3	poset ("Statistical Pale")
	posit (solar of describe ())
	15 - 114 1 column larger and collected (*)
#4	part "Unive label for column 'octan-presents; ")
	port (sales -de l'ocean position 1. univel)
	point (v/n²)
	post ("Atoristas (columns) en deterat have moreto value court
45	Print (Prints (Rumas)
	notice = sales -d-unal () sun ()
	Aftern = salls - at the marks value = windle (missing 50)
	columb with missing values (missing so)
	1340 (States And 1929)
	point (* lot)
1.0	0 1 1
OP12	Sample date: Tenstude latitude houring-metan-age total process total between
	192.93 345R 41.0 880.0 199.0
4.1	population hausheds mation income median have value correspond
1 10	799.0 126.0 8.39.52 459600 NEAR BAY
- 27/13	It would be not write more
	Statistical No : PTO
	The state of the s
	Joseph State (4)

	A PAGE DATE . ,
	phone of the columns on adultacer & diabetes were have null values
17	Checked using of a israel sum()
//	Handling Method
//	1) If missey values are low, they are dry 1
//	Missey values are neplaced with allowed using of a dropping)
//	Handing range of all of they are dropped using of a dropped () Nissen values are neplaced with column mean of a filling (dfa.
//	(alegastal columns are filled with most frequent rate at - a fillna (af - a model), loc (o), emplace - true)
//	d-a- Islna (d-a-mode), slar (a) enders to
	mpare one
	Exacting categorical columns
2	In Diabete datasel, there was no categorical column In Adust datasel, categorical columns are workclass, columns, marchal status occupation, relationship,
	Doubt depret colemna de la comma
	In the solvention operate of the
	warkens, concern, martin land occupation, relationship,
	sace, genous, nature-country, encome
	11 2 2 1/110 1
	from extraor proprocessing emport Label Encoder Je = Label Encoder()
	le= Labelt nooderl)
1	df-d ['gender'] = le-fil-tranform (df-d ['gendor'])
2)	Orly In Men Max Scaley & Standondization
4	Feature Men-Max Scaling Standardization
	Religion Scales value 5/n 0 & 1 Centres value assurd man & std=
	x-scale = (x-Xmin)/(xmax-xmin) X stated = (x-mean)/stated
	When to use 9 when data is bounded & needs when data has nowned other &
	Effet on outliers Sensitive to outliers Less sensitive to outliers
	Special on gullier Sensitive to pullions less sensitive to outliers
-	(buoz it depends on nin/nouva) (centers data assum no
	the state of the s
	Drabetes: Standardisation was applied since medical data lite cloud-poses
	glucose-level, etc may have outliers.
	Adult: Non-Max scaling was used to normalise growe, a
	E house per-neek
1001	100- 100 miles to high 1
W/AM	. 199
V.	1303 - 1270 5 2 mm / 100 5 10 mm

```
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.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"]
```

from sklearn.compose import ColumnTransformer

```
# 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

bee	Date
lab - 4 linean Regression lab - 4 linean Regression	4) Quatabution plot - Victimity 5) Realismship b [w variable 6) Split the data 7) Train the model 7) Quadrit the election
80-1/-18	10) Values of Constitution

history Oceanican	Pope
linear Regression	1. O/P- Admir political a bland
i ant and a a of 8-5-	Predicted - per capita income 2020: \$ 11288.69
import pandas as pd	
import numpy as no	2. Off-
import matplotub. Pyplot as plt	preduted salary for 12 years of experience: \$139043.67
from sklearn model - relection import train 1	# Multiple Regression
from skleam linear model import linear regre	The state of the s
from sklearn, matrix import mean-squared	import numpy as pa
2 = 50	in and hatteril wilt will
lile path = " Content Canada - pon - capita - in	income csv impart matplotlib, giplot as ptt from sklearn linear model impart linear Regression
di-nea	- Just Skeavit, Milean - Mood Ampan umanikagawan
de = od need csy(lile_mth)	City noth - 61 Content I biging ony" - 910
df. columns = ["year", "pin_capita_iniceme	gile path = " (antim hining . 55"
ag. courins - L span ,	df = pd. read _ csv (file - path)
14 FG 8 1177 Historial to	df. loc [: " vest ward (out of 10)"] = of [" tut woul
X = df [["year"]]	out of 10)"]. Julyac af ["Text_use ("out of 10")] x
y = at the cupies - where	the state of the s
model = linean Requision () model. fit (x, x)	2+0 (3
madel (+ (x x)	all contains - representative
predicted income 2000 - model predict (if initance (val, tta):
andited imcome = 2000 = model. preduct	((2020)]]
Island tal	
plt. water (x, y, color = 'blue', label = ')	Return Data : return mand to num. get (val lower (), 0)
plt. scaller x, , total color = 'ned';	label = Kig
at olat (x, model-place	Will will be a second of the s
plt. Mabel ("year")	
ptt. yabel (" pen apita income")	of loc [:, experience] convert experie
pro- yatra	de dropna (implace = True) " "tot score (out of 10)",
ptt. legend ()	\$ Spredictio X = of \$1 " experience", "tot work (out of 10)",
print (f" Predicted for repita mismo 2020:	+ Condition x = of [1 " experience . Cost of 10)"]
it with minme 2020:	* interview - Mare Court of
soint (+" Preduted per repense	11 La Corcology (B)?
model = Jinean Regruien	\$ { predicted
model = linear Regression model = linear Regression model = linear Regression cardidates = pd-Datafrana ((£2,2,6), [12,10,10]) columns = ("esponimes", "tut user (sut of 10)") far i, salary in snumerate (predicted solarius); print (+" bredited salary for landidate \$1+4; \$ salary = 2+19,	3. Enedicted Salary: \$ 36,000 Yes, 3 muched categorial value like state using one not Freeding to comment the categorial values into kinary vaniable.
model = linear Regresser linear Regresser linear Regresser linear Regresser model = linear Regresser linear Regresser model = linear Regresser linear Regre	3. Credited Salary: \$ 38,000 Yes, & meeded categorial value like state using one ust Friending to comment the categorial values into kinary vaniable.
model = linear Regresser model = linear Regresser model = linear Regresser model = linear Regresser cardidates = pd Dataparae ([[2,3,6], [12,12,12]]) cardidates = pd Dataparae ([[2,3,6], [12,12], [2]]) (churms = ["espanaes", "att was (aut of 10)") for i, salary in trummont (predicted salarius); print(+ "Pudited salary for (andidate 1: \$185.11) Colp = 1. Predicted salary for (andidate 1: \$285.11) Predicted salary for (andidate 2: \$5.44.86.23) Redicted Predict: \$5.105.30, 93	3. Credited Salary: \$ 38,000 Yes, 3 meeded categorial value like state using one ust Freeding to comment the categorial value into kinary vaniable.
model = linear Regresser linear Regress	3. Credited Salary: \$ 38,000 Yes, & meeded categorial value like state using one ust Freeding to comment the categorial value into kinary vaniable.
model = linear Requirem model = linear Requir	3. Credited Salary: \$ 38,000 Yes, & meeded categorial value like state using one ust Freeding to comment the categorial value into kinary vaniable.
model = linear Requirer model = linear Requirer model = linear Requirer model = linear Requirer cardidates = pd. Datafrane (CE 2,3,67, [12,10,10]) columns = ("espaine", "att. ware (out of 10)") far i, relary in enumerate (predicted solaries): prival (+" breated solary for landidate \$1+13; } Solary: 2+1") Off Cordital Solary for Candidate 1: \$ 38581.11 Predicted Solary for Candidate 2: \$ 54486.23 Rudital Profit: \$ 5105.70.99	3. Condited Salary: \$ 96,000 Yes, A encoded categorial value like state using one ust Encoding to consust the categorical values into binarry vaniable.
model = linear Regression linear Reg	S. Pardicted Salary: \$ 96,000 Yes, A meeded categorical variar like state raining one- not Friending to common the integrinal values into binary vaniable.
model = linear Regression linear linear Regression model = linear Regression linear	Sendicted Salary: \$ 36,000 See, & invoced categorical values like state using one-wat Enroding to consust the integrated values into binarry variable. The sending to consust the integrated values into binarry variable. The sending to consust the integrated values into binarry variable. The sending to consust the integrated values into binarry variable.
model = linear Regression linear Reg	Sendicted Salary: \$ 36,000 See, & invoced categorical values like state using one-wat Enroding to consust the integrated values into binarry variable. The sending to consust the integrated values into binarry variable. The sending to consust the integrated values into binarry variable. The sending to consust the integrated values into binarry variable.
model = linear Regresser lize [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2]	3. Condited Salary: \$ 36,000 See, a invoced categorial value like state using one-ust Encoding to consust the integrated values into binary variable. The salary is a salary to the integrated value into binary variable. The salary is a salary to the integrated in the salary into the salary int
model = linear Regresser lize [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2]	Sendicted Salary: \$ 36,000 See, & invoced categorical values like state using one-wat Enroding to consust the integrated values into binarry variable. The sending to consust the integrated values into binarry variable. The sending to consust the integrated values into binarry variable. The sending to consust the integrated values into binarry variable.
model = linear Regression model = linear Regression model = linear Regression cardidates = pd. Pataframe ([[2,3,6], [12,12,12]]) cardidates = pd. Pataframe ([[2,3,6], [12,12], [0]]) cardidates = pd. Pataframe ([[2,3,6], [12,12], [0]]) [an i, solven in trummont (partitle salviss); print(f + Partitle salven for (partitle salviss); print(f + Partitle salven for (andidate 1: \$ 198581.11) Predicted salven for (andidate 1: \$ 38581.11) Predicted salven for (andidate 2: \$ 51486.28) Rudited Prefit: \$ 510590.99 See 1 performed data provising stops like Handling raising values and encoding categorical data to must be described and por modeling. See 1 rusualized the regression line.	Ses, a smooth categorial value like state using one not Encoding to consunt the sategorial value into bonary vaniable.
model = linear Required " and linear Country model = linear Required " and linear Country " and linear Country for and late of linear Country model = linear Required " and late of linear Country for and late of linear Country linear Country model = linear Required " and late of linear Country linear country model = linear Required " and late of linear Country linear Country model = linear Required model = linear Required model linear Country model linear Required model linear Required model linear Country model linear Required linear Re	3. Cardited Salary: \$ 36,000 Yes, & invoced categorial value like state variage one-not Encoding to convert the integrated values into binary variable. The salary of the integrated values into binary variable.
model = linear Regresser linear = linear Regresser model = linear Regresser model = linear Regresser linear = linear Regresser linear = linear Regresser model = linear Regresser linear = linear line	Sendited Salary: \$ 28,000 Yes, & invoded categorial value like state rainal one with Encoding to common the categorial value into binary vaniable. The control of the control of the partitioning to be control of the

```
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

		SN
lab-	30 P : porter latel	
		Dute Page
1) logiti regression eq	0.8	Softmax $(z_1) = e^2$ $e^2 + e^1 + e^2$
categorial where mital hances	Till tommer of tricken	Softmax (2,) = e2 0.65 month
1) logitu regression eq	vation	e2+e1+e°
		hy is school trapial
$p(\alpha) = \frac{1}{2}$	= 1	Softmax (22) = e' = 00044
1+0	1+e-(-3+0.80	The sport of the terms of the state of the s
is col - 1 lift ut	T11 1 1	
ii) Calc protability that a	sucdent who studie	Softmax (23) = e = 0.091
+ nu mui pass		Little College world with the time of time of the time of time of the time of time
-122 1		probabilities of the 2 class and appace is st, 2.
$\rightarrow x = 7 p(x) = 1$	-(-5+0.8(7))	photodian of
		- Co 03"
= 0.	6457	C til-x
iii) Retermine the graduited	1 0-1 1 1	from the same well graped begate sugaries
based on threshold of	class (PF) for this	June 1 - Dental Regulation (1)
ausin on investing of	0.5.	Calculate a salate of the house
> PCx) = 0.6457		hit y
p(x) 2/ 0.5	^	147
	y = \ 1 ij P60)>,	co (the religion to better a
Thus y = 1 (pan)	10 otherwise	has been
The state of the s	de l'astronial	tandini lihar
Carriely 2 = [0,10]	Trible Blocker	
Comider 2 = [2,1,0] to find probabilities value	for three claves apply	se chare trapin
to find probabilities value	s of 3 claves.	100 hianjin lab
	ð	Cle Age don to 1 1 million
Softmax (212) = e	41	
Softmax $(2i) = e^{i}$	e 2j	the periodice for (1911)
	2 12 12 12 12 12 12	Carp Indiana
		CAL THE COLORS
		a Junite

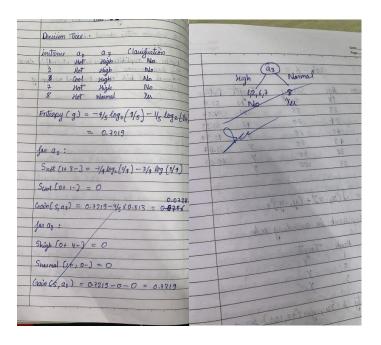
	100-3	Don't Laborate the second seco
>_	Binary Logitic Regnuism	098 = 35
		age = 35 prediction - Sanction (age)
	import pandas as pd	- Cago
_	truped panda as put them matplotlib impart puplet as pit	0/9 - 0.3709834
	d. pd. read _ CSV ("Horium) prosent	
-		DOLL To copy don mercun
	of real (de age, of bought inwamer, marken = 12)	Mutticlass logittic Regression
		1.1.1
10	x Town . Y Test 11 Town . II lest = main test splet of the	umport pandas as pd
	D, df. bought_incurance, Tanin_ gire = 0.3, random this =10	from skleans dataset import load inic
		William Skullyla Indones - Muchan Jurgan Blass State
	x-Tut	from Sklearn linear model import logistic Regression
	from stanen Jimen model impart logitic Regressen model = logitic Regression ()	from sklean metric import accuracy score
	model = logitic Regression ()	from skleaam import metrics
	model . fit (x train, y train)	
	X-Ted	inix = pd. read -csvil "/ Content/inix dsv ")
	y_tid	
	x-predicted = model. predict (x-ted)	X = inis drop ("species", air = "columns")
9	y_predited	y = inis, deep (ipinis, our = toumbs y = inis, ipinis y toum x toum x toum tut split
	model coef-	g = var. sparse y to aim y test = toain Test - split
	model_intercept_	y = iris. quille X _ train, X - trid, X _ train, X - trid = train _ trid - split X _ train, X - trid, X _ train, X - trid = train _ trid - split X _ train, X - trid, X _ train, X - trid = train _ trid - split X _ train, X - trid, X _ train, X - trid = train _ trid - split X _ train, X - trid, X _ train, X - trid = train _ trid - split X _ train, X - trid, X _ train, X - trid = train _ trid - split X _ train, X - trid, X - trid - split - split X _ trid - trid
		A CONTRACTOR OF THE CONTRACTOR
	import math	2 Coulti day = multinomia
	def rigmoid (2):	11 - logulu kayımı
	return 1 ((1+ math. exp(-x))	
		model and the tid
	del pondition-fune (age): z = 0.12 + rage - 4.973	model for (c. town, graded (c. tod) y ford = model, product (c. tod) uwaay = awagay were (y tod, y - pred) uwaay = awagay on the tod st : {awaay 21}
	Z = 0.127 * age - 4.973	accuracy = accuracy
	y = sigment (2)	the tool set : Sauvary: 27
	situan (y)	sign (+ sauray on the ten
	· ·	paint (15 hunay on the tot at ; (away 21) Conjuien mateix = ruther conjuien mateix (y-tot)
7		retries confusion mains of
		Cansulan + muun

	3 -4	_350
cm-display plot ()		Dote J Pope
It. show ()	ردننى	There were no mining or inconsistent values
	dii	The Confusion matrix showed achieved 98.24a
curacy on the train at: 1.00	Civ vi)	Reptile and amphibians have mostly minianified down
by so rabing togic		They share reminent statures. Memmals and birds also as they had our expering
IR- Comma - sep. (SV you distall months and		family toll 8
tugation level: lower ratingation - pighen turnous		The state of the s
one spent in company: longer tenure is higher library		8105.0 =
land a law colony - higher attaintion		Ju as :
continued: Sala 4 tehrnial nave higher turnous		(4/2) post file - (4) post file 2-2 est tone
the sound of the s		0 = 6-1 10/12/5
it is a good accuracy, but accuracy alone can richading due to potential class imbaldone.		600 C 0219-46 X 0 612 - 6022
(court y with 1 th war	_	1.0000
o dataset		0 2 (-4 10) 400
, pengament data properties of steps.		0 = 1 = 0 = 1 = 1
and the animal name column. Standardized the grater	-	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
g itandardsgaler to improve model performance	1	P1810 0
	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		
LACE LACE LACE LACE LACE LACE LACE LACE		
import panda		
impert pander as pd. from Steleam tree import Decision less Claufen from Steleam vertices import their		6000 Œ do
from Chippon with the import Downson Tree Claviller		0 3 0 0 0
from Skleam metrics import train tell split	Actual	00500
Import reaborn as some	-	000110
impost matplotlib pyplot as plt		10000 15, Jan 18 18 18
		defeat for implet Principal States
df = pd . Trad csv ('/ lentral / drug .csv ') X = pd . Trad csv ('/ lentral / drug .csv ') 4 = dl (') Drug '?		Predicted
X = pd. get durinies (df. drop (Prug! air = 1))		and similar or care the
y = of [orug]		the intelligible of the telephone to
Post man when some server server		ole -
X-train, X-test, X-train, X, test = train_test_split(x, y,		C very punch (mitted) I very horn, by - by
tut_cize = 0.2)	A CONTRACTOR OF THE PARTY OF TH	Mean absolute error: 57-6
dtree = Decison hee Clariling	- 12	lean squared erron: 6338.2
dtree = Decison Tree (Varifier () dtree = jit (x - train, y - train)	- K	not mean equared error: 79.6128
u aned = Atree and to y test		ve training of the V. Town V. Let - Francis ted as
y pred = dtree, predict (x-text)	- 43	Co - grip tak
1 11 11	fola Ac	wrony was 1.0 for in datard
accuracy = accuracy - none (y-ted, y prod)		d count u inst y) is and
Deuray = acuray - More (y-tid, y-pred) paint (+" Preuray : (acuray)") cm = confusion matrix(y tid, y-pred) plt. (gure (jigsipe = (8, 1))	11	he Confusion matrix enly has diagonal eliments, del made zero errors.
cm = confusion matrix (y tet, y-pus)	me	del made zero errors
nlt. liaure (signize = (86))		Charles to the manual of the m
ott charles	00	S THE STATE OF THE
pt. show ()	1 Van	etter is to the state of
	- Ru	realistics like population density and awage in
0/6 —	ant	scatures like population density and awage in
		Con Comp to
Accuracy: 1.0		
		0/0
1001		11/2
		hamary 1.0
		O .

```
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.

A [8 500 N 51.3] B 23 55 N 41.57 C 24 30 N 51.35 D 41 60 X 10.44 E 43 70 Y 31.04 F 38 40 Y 60.09 X 35 100 ? Distance = J(\pi_0 - \times \frac{1}{2} +	Reason	Age	Salary 50	Tanget	Distance	Lab-6 topp minimum
13 23 30 N 3135 C 24 20 N 3135 D 41 60 Y 1044 F 43 70 Y 31.94 F 38 40 Y 60.09 X 35 100 ? We shawn model neighbour impact town the split from Schann model neighbour impact Knighbour leviglar inic data = pl. viad s V ("via data csv") X inix = inix data ibe [:, -1] X inix = inix		18			52.8	The tent of the tent
D 41 60 Y 1044 E 43 70 Y 31.04 F 38 40 Y 60.02 X 35 100 ? Distance = \(\tau_0 - \times \) \(\tau_0 - \ti	B					William Onnday as all
F 38 40 Y 60.09 X 35 100 ? Dutame = J(To-X) + (yo-y) + Pan K = 3, namk in aunding order Outame Rank Tanget 31.95 2 40.94 3 Since majority is Yes, Jan (95,100) tanget will Comp matrix = leng fraink (y tid y ord risk) Since majority is Yes, Jan (95,100) tanget will Open (class product of the p		24				impail realborn as sns
X 35 100 ? Ditame = J(x_0-x)^2 + (y_0-y)^2 +	0	41		1000	40.44	from skleam model Water in the
X 35 100 ? Ditame = J(x_0-x)^2 + (y_0-y)^2 +	E	43	70	у	31.04	from Schann model neighbours in the split
Distance = D(To-X) + (yo-y) + Since majority is Yes, for (\$5,100) tanget wills Print for majority is Yes, for (\$5,100) tanget wills Out a gray some: 1.0 Genfauer: 100 This data = pd. read sv ('min data .cev') X inter majority is Yes, for (\$5,100) tanget wills Open to the point of (y int y post int) Open to the point (y int y post int) Open matrix = least matrix (y int y post int) Open matrix = least matrix (y int y post int) Open matrix = least matrix (y int y post int) Open matrix = least matrix (y int y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y post int) Open matrix = least matrix (y int y y y y y y y y y y y y y y y y y y y	F	38	40	У	80.09	KNughtour Classifus
Quitance = $\int (x_0 - x)^2 + (y_0 - y)^2 + \dots + \int (x_0 - x_0)^2 + (y_0 - y)^2 + \dots + \int (x_0 - x_0)^2 + (y_0 - y)^2 + \dots + \int (x_0 - x_0)^2 + (y_0 - y)^2 + \dots + \int (x_0 - x_0)^2 + (x_0 - x_0)^2 + \dots + \int (x_0 - x_0)^2 + (x_0 - x_0)^2 + \dots + \int (x_0 - x_0)^2 + (x_0 - x_0)^2 + \dots + \int (x_0 - x_0)^2 + (x_0 - x_0)^2 + \dots + \int (x_0 - x_0)^2 + (x_0 - x_0)^2 + \dots + \int (x_0 - x_0)^2 + (x_0 - x_0)^2 + \dots + \int (x_0 - x_0)^2 + (x_0 - x_0)^2 + \dots + \int (x_0 - x_0)^2 + \dots $	X	35	100	2		inix data = ad next an China
Distance = \(\tau_0 = \times^3 + (y_0 - y)^2 + \tau_0 \ta						Y inis = inis data (cv)
for K = 3, name in auending order R = 3 Name	Technolo.	Calls	-36 km	The same of		V day - min 1 to 1 to 1
Some majority is Yu, for (\$5,100) tanget will sprint (for Anyrog Score: Sacuracy 3") print (class majority)	Distance	- 1/2	-x72+(11-L	1 12		v to
for K = 3, rank in aunding soder R = 3 Outame Rank Tanget Since Rank Tanget Since majority is Yes, gar (95,100) tanget will law report (10 tanget) Since majority is Yes, gar (95,100) tanget will law report (10 tanget) Privat (First August Sacre Sacrary 2") Privat (Can report) Olf — Recuracy respect (2) Genfaure : 10 Genf	- Lunumi	- Vido	and and	10'		- Man mus, X-tell wis, Y-test-wis (X-wis, Y-vis,
Distance Rank Tanget 31.04 31.05 2 Y 40.94 3 Y Comf matrix = Conf inative (y tota y y paid inic) Since majority is Yee, you (95,100) tanget will private (for Annuray score: Sacuracy ?") privat (comf matrix)	0 - 11 - 1	2 0.		· a code		teit_size = 0.2 , nandem -stati=
Dutame Bank Tangel 31.04 4 31.95 2 40.94 3 Since majority is Yes, gon (35,100) tanget will then report (10 tanget) and	JOI K 2	s , scan	y in cuienai	my will	1000	
31.95 2 Y 40.94 3 Y Since majority is Yes, Jon (95,100) tanget will from report = (lampiature supertly tid, y pred in 1) print (fin Augurez supertly tid, y pred in 1) print (com matrix = (lampiature supertly tid, y pred in 1) print (com matrix) print (com matrix) print (com matrix) print (com matrix)	۸.۱-	a 1	+ +		1100	- Knn_clarifier = KNeighbours (n_neighbours = k)
31.95 2 Y 40.94 3 Y Some majority is Yes, gar (95,100) tanget will the report clausfeation majority is Yes, gar (95,100) tanget will the report clausfeation majority is yes, gar (95,100) tanget will private (for majority majority in yes private (for majority majority) private (for majority) private (form majority) private (form majority) Ole — Recuracy score: 1.0 Genfauren: 10 0 0 9 0		-	Tangu			- Knn - classifier . fit (x train inis, y train inis)
accuracy =	100000000000000000000000000000000000000		- У	De Taring	11/1/19/19	to find but & ted multiple Solve & play the course
Since majority is the, for (\$5,100) tanget will from report = (law finition report) it is, y and in yes. Print (f" Anunery score: Saurany ?") print (clau report) Ole Accuracy more: 1.0 Genfavior: 0 9 0	31.95	100	У			- accuracy = accuracy - were (y-teit_iru, y-pred-low)
print (fir Arunacy score) sauracy 2") print (clau meant) Ole Accuracy more: 1.0 Genfavion: 10 0 Genfavion: 0 9 0	40.44	3	X/			Complementaix = (and matrix (4 text, 4 pred inis)
print (fir Arunacy score) sauracy 2") print (clau meant) Ole Accuracy more: 1.0 Genfavion: 10 0 Genfavion: 0 9 0						- I mater combination report (4) tet, 4 pred in
print (fir Arunacy score) sauracy 2") print (clau meant) Ole Accuracy more: 1.0 Genfavion: 10 0 Genfavion: 0 9 0	Since mo	instu is	yer. Jan (85.100) to	mant will	Man super stranger
print (clau report) Of P Becuracy score: 1.0 Genfarion: 0 9 0	IIII.	10/	, 8		0	2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1
Of P Becuracy More: 1.0 Genfavin: 10 0 0 0 0 9 0	9	/	1			print (+ "Arway State)
Of P Becuracy More: 1.0 Genfavin: 10 0 0 0 0 9 0		ON			1910	print (conf matrix)
Of P Becuracy More: 1.0 Genfavion: 10 0 0 0 0 9 0	A 100 (10)	7				orient (clau-report)
Gnfauim: 10 0 0	37 37					
Gnfauim: 10 0 0						The state of the s
Gnfauim: 10 0 0	Name of the least	LANGE BALL	LEGIL			do 12 G M
Gnfauim: 10 0 0						011
Gnfauim: 10 0 0		ALL DOOR	23219 mon			Accuracy Mail
				100		
	100000		100000			Conlavion: 0 9 0
				1		

			Date Page
10)			
Classification	n Report	106 6	
10.00	Precision		1- MOTE superit
Setora	1.00	1.00	
vericolor	1.00	1.00	1.00
Virginica	1.00	1.00	1.00
	HOO /	100 la	1.00 30
auracy	1.00 .		1.00 30
mairo aug			1,00 30
wiighted aug	1.00		
N min V	win tot Y	with the y	X. tagin inix,
Steam Charles	- sie tit	Sale Bridge	
flan Street	d some	6936.2	8 = 3
K = 2 - quie	x 1000 lo au	curacy here t	ut generally tak
K TE ADDR	no nata = 1-	according = 0	o (no mirchain
+ 11 1 1-	to tet m	High Salue 4	plot the error of
	I A AKIA I'L	mayor vans	par, su como
to find the	to down the last		
to find the	tot day go	ANNE ALTO	
Carol his	the purpose	u Vintage To	Con water = 0
H ù needed	because	lialuze have	diff ranger (eg
H ù needed	because	lialuze have	diff ranger (eg
H ù needed	because	lialuze have	diff ranger (eg
H ù needed	because	lialuze have	diff ranger (eg
H ù needed	because	lialuze have	diff ranger (eg
H ù needed	because	lialuze have	diff ranger (eg
H ù needed	because	lialuze have	diff ranger (eg
H ù needed	because	lialuze have	diff ranger (eg
H ù needed	because	lialuze have	diff ranger (eg
H ù needed	because	lialing have water hard sustained day	diff range (ig uahr mune ig a. Amproves M after making)
H ù needed	because	lialing have water hard sustained day	Carl matrix = 0
H ù needed	because	lialing have water hard sustained day	diff range (ig natur mune ig ā. Impreses st ofta. maling)

```
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

177 - 1714 - 1714	Page.
181 = 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181 181	
Orous an optimal hyperplane using times sum to classify the following points	$\alpha_{1}(6) + \alpha_{2}(9) + \alpha_{3}(9) = +100 \alpha_{1} = 13/4$
to clausly the sollowing prints	$x_1(9) + x_2(6) + x_3(9) = +1$ $x_2 = 13/4$
00 + 10 ch + 3 ch = 10	x 1(9) + x2 (9) + x2 (17) = -1 x3 = -7/2
(1,1)(2,1)(1,-1)(2,-1)(4,0)(5,1)(5,-1)(6,0)	Marify the Solliening points
(0) \(\chi_1\chi_2\chi_3\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1\chi_1	$\omega = \chi_1 \widetilde{S}_1 + \chi_2 \widetilde{S}_2 + \chi_3 \widetilde{S}_3$
(1) (2,1) (1,-1) (2,-1) } +ve	$= \frac{13}{4} \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} + \frac{13}{4} \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} + \frac{13}{4} \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} + \frac{13}{4} \begin{bmatrix} 4 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$
(4,0)(s,1)(s,-1)(6,0) 3 -ve	- 1/4 + 13/4 - + -1/2 0 4 = -1
V = 100 28 5 4 200 00 27 8 5 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4	34- 8 (0.8) (1.3) (1.3) (0.8)
produced the way of the production	= [1]
0 = +4	= - []
and the second of the state of	2 = [+1] 1 =-3
in to some of the sound	$2 = \begin{bmatrix} +1 \\ 0 \end{bmatrix} \begin{cases} b-3 \\ b+3 = 0 \end{cases}$
	intercept on x-axis = 3
-5 -4 -3 -2 -1 1 6 8 4 5 6	2(1) - line parallel to y-air
2	(0)
3	
	Sor of
	\$.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 second to be such that the second
St = [1] -2 (-1)	Commence of the latest and the lates
(2) S2 = (2) S3 = (4)	
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	1 = 65 [3] = 85 [5] =
23-4-55 = +1	
	0 = 0 1 2 2 2 2
α12125 + α22223 + α3 2322 = -1	
1 5183 + 203833	1+= 33,0+37,0+35,
	1 + = 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

Code Code	limear sum auway: 1.0 - mis
Code Sum	Confusion matrix has a solution trapped
jam sklaan medd slutten impest train tid split form sklaan surn impest svc. form sklaan mitus impest svc.	10 0 0 112 topon with many many many many many many many many
many Jase, tenjungar	in at = phound cey (" in ce ")
inix_df = pd_mad_csv("inix_csv") X = inix_df _drop (lotum ns = [sprinx ']) Y = inix_df ["xprinx"]	Guitani = 20 may 2 park 1 min = 2
(x, y, ted size = 0.2, norder state = 12)	Both RBF and brican gout the same accuracy was
svm_nbf = svc (kund = 'nbf', nandem_ital = +2 svm_linean.fit (x_train, y_train) y_pred_linean = svm_linean.ppdit(x_trd)	The AVC Mare was 36 - 38%. It shows that I
	AUC model performed pretty will for the letter -
print C"RBF Acuracy":, surray-rese (y tist, y pred not))	The AUC MORE FOR JON ING. CSN MAS 1.00 ON IT
print ("RAF Acuracy"; surray - nore (y tot) print ("In linear giverary ", gueracy riore (y tot) prid north linear	way small and clim dataset with well apparation co
NP -	7.8. 90
RBF SVM anurary: 1.0	SPE Strikt amount : 1.0
Confusion matrix -	ratea materi
0 9 0	0 0 01
0 0 11	1100

```
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

-	1	Date	Done Page
	lab=8	makers for authorities or	-> Algorithm of random Forest
	Difference between Decide	or tree and Randors Farest	1) Tripit datasit O with fatures & dass labels 2) Sit the no. of trees no otionates 3) for each true is from 1 to no estimators.
-	model type is single	Model type is ememble of	really tripe is rough . Model tripe to man the
- 100	Prone to oversetting	les prone due to ownaging	Code:
3)	complex data	accuracy higher due to ensemble learning	from sklean model whiten import train ted with from sklean mounts import Randomform (laufin from sklean mother import auricus wege
+>	Factor to train and predict	Slower due to multiple trees	fire = pot read esv (ini csv) Don't at interest
E)	low bias, high vaniance	low bias, reduced variance	they bine, high undiance how hors, ardund you
->	Panameters of RF	- Analyst a Ayes Mr	oc brain, oc ted, y train, y ted, = train. tet. (2/4)
(1)	n_estimation — No . o Clan_wight — whight of max depth — max dep bootstrap — whether boo	mociated with clause of each tree Istrap samples are used when	of default = Randomport Clarifier (h_ utinuter 10, navtern- rate = 12) If default of Cx-train, y train y-pred default 0 = / ef. default-predict (or tate)
	n jobs - ne of jobs nandom state - med	to run in parallel for reproducebility	print Causey defaut : 47) where (y test, y pred) where the constant of 10,50,100, 200, 500]
,	unbose - controls the	unbosity of output	print (sucru)

= 40:1	But n- eitoraton: 10 but-aumauy wan: 1				Page
Best n- etwater: 10 best assurance un : 1	Best n- esternation: 10	ofe - away w	ish n-edi	matan =	10:1.0
But n- aunaug una 1	But n- aunauy - unn : 1	at several die			-100:
but annay was I	but aunau duri il	Best n- estimator	:10/	a labour	300,
The state of the s	Manual III with the state of th	but-auracy - M	100:1	MARKE NO	
County the parents from the parents of the parents	County the particular of the June 18 is a start of the June 18 is a st		90		- interes
Count store to make the store to the store of the store o	County long a cathod for the party of the pa		10000	1 713	-
Manufacture of the Joseph Company of the party of the par	A day of the James the same of	The second second	- labour	-	Timoten
The state of the format of the state of the	The spirit of the form the spirit of the spi	record that to	noted tras		The table
A STATE OF THE PARTY AND A STATE OF THE PARTY	A to have the and the form		A Note to the	nous cal	The ora
the plant to 2 th a classed or part of	the short to 2 the short of party to			-	
A Talland Talland of part of	Columbia de la columbia				
the plant of the p	the house to a the along to be the		200		
the plant to 2 th a classed or part of	the short to 2 the short of party to		TE NEVA	The Toll	To their
in laboration with a laboration of the sales of	is to have the same of the same	311000			A CHI
the idea	to ide a		W 10-11	and the	Jud de
- Control of the cont		the land to a	11-1-11-11		-
		9		220	ala hu
				Marie Control	

```
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

1	Date Page
•	
73	1ab - 9
	1 : 5P =
	Booting: Combines multiple weak learners to south
	a thing learner. It works by training
	model argunitally where each model for
	model significally where each model for on errors model by premous one.
	Parametiri:
	estimator The base model
	n utimatar no of weak learner
	dearning note shrunks contribution of each learner
-	algarithm SAMME.
	sandom-rate for suproducibility
	Alao ·
110	Algo:
,	Start with equal cots for all training sample
	Story with selection to so do not with the selection of selections
0	Tagin a week made
2.	Train a weak model
2. 3.	Train a weak model Cal war 4 update sample with
3 . f.	Cal wron 4 update sample with a wt band on
3 . f.	Cal wron 4 update sample with a wt band on
3 . +.	Cal wras 4 update sample with a wt band on aurany. superat n estimater
3 . +.	Cal wron 4 update sample with a wt band on
3 . f.	Cal wras 4 update sample with a wt band on aurany. superat n estimater
3 . +.	Cal wras 4 update sample with a wt band on aurany. superat n estimater

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

	140-10
	K-Means
	Partition data into k clusters by:
1.	Randomly initializing untroids
2.	Assigning points to the nearest gentlined
3.	Updating rentroid based on arrianed point
4.	Randomly initializing untroids Assigning points to she nearest aentroid updating rentroids based on assigned points Repeating until sonuergence
7	Comment of the Commen
->	Choosing number of clusters
	Elbow Method
•	Silhouette Score
•	Elbow Method Sithouette Score Domáin Knowledge
>	Sum of sound epopers
	Sum of squared errors $SSE = i = 12 k x E Ci 2 x - ui ^2$
	Key Panameter
	n cluster No of dusters to germ
	initialization method
No.	n init No. of initialization
	max deration per num
	Control nandomner
	Convergence Mireshora
	algorithm algorithm to use

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

PCA 1. Calculate Mean 2. Calculation of Covaniance matrix 5. Eigen values of covaniance matrix 6. Computation of the eigenvetor - wint eigenvector 7. Computation of 1st principal components 8. Cremetrical meaning of first principal components 8. Cremetrical meaning of first principal components	$(\operatorname{ev}(X_2, X_1) = -11 = (\operatorname{ev}(X_1, X_2))$ $(\operatorname{ev}(X_2, X_2) = 23$ $(\operatorname{ev}(X_1, X_2) = 23$ $(\operatorname{ev}(X_1, X_2) = (\operatorname{ev}(X_1, X_2))$
B. Finture ex 1 ex 2 ex 3 ex 4 x_1 4 8 13 7 x_2 11 4 5 14 x_1 1 4 5 14 x_2 11 4 5 14 x_3 14 5 14 x_4 15 14 x_5 17 1 8 18 18 18 18 18 18 18 18 18 18 18 18	Eigenvalue of Consense matrix $0 = dit (s - 2t)$ $= \begin{bmatrix} 14 - 2t & -11 \\ -1t & 2s - 2t \end{bmatrix}$ $\Rightarrow 2^2 - 3+ 2 + 201$ $2 = 1 (3+ t \sqrt{5}t6)$ $= 30.3849 \cdot 1.6151$ $= 2 \cdot 2 $

						No. of Belleville
T					Dote	
3	lli =	. 112				-
	n	(14 - 7v)	DCI		3	
	de 1 =	117, 1	12 - (14-	2,)+.		
						-
-		U, = [118	-			
		L 14-	71			
TO 8	and wi	remuster		+ 1		
1	6	gen vestar,	we compo	le lingth	-of x1	
	11 11	111 = 11	112+(4-2	72		
		= 1	9.7348			
100						
	14	nit eigenree	ctar			
	10	nt eigenver nuponding	to 21	01 = [1	/simil.	- 38
	-	- 7		LC	14-2,)/	11 4, 11 _
C2	= 0	- X30 3				
		. 331.	10.			
	+at	of 104 pa	inical .	propagen	t	124
Comp	waum_	9	initipate a	7		
	TOCOL					
	DC2H X211					
					20	-
oT [XIK -	x, 7 =	[0.5574	- 0.830	XIR	- 1
1	× 24 -	X2	-	/	Xzk	- X2
					1 0 00	03 (X2K
		=	0.55 74	XIX -XI	1-0.83	~
			The same	-	Je o	19-00
				_		
				-		
						-

```
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}")
```