# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.



## LAB REPORT on

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

Submitted by

Mohammed Zeeshan Umar (1BM22CS160)

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 Mohammed Zeeshan Umar (1BM22CS160), who is bonafide student of B.M.S. College of Engineering. It is in partial fulfillment for the award of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements in respect of an Machine Learning (23CS6PCMAL) work prescribed for the said degree.

Lab faculty Incharge	Dr. Kavitha Sooda
Name : Amrutha	Professor & HOD
Assistant Professor	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	15
4	17-3-2025	Build Logistic Regression Model for a given dataset	26
5	24-3-2025	Use an appropriate data set for building the decision tree (ID3) and apply this knowledge to classify a new sample.	30
6	7-4-2025	Build KNN Classification model for a given dataset.	35
7	21-4-2025	Build Support vector machine model for a given dataset	48
8	5-5-2025	Implement Random forest ensemble method on a given dataset.	57
9	5-5-2025	Implement Boosting ensemble method on a given dataset.	61
10	12-5-2025	Build k-Means algorithm to cluster a set of data stored in a .CSV file.	64
11	12-5-2025	Implement Dimensionality reduction using Principal Component Analysis (PCA) method.	71

### $\textbf{Github Link:} \quad \text{https://github.com/ZeeshanBMSCE/1BM22CS160\_ML\_LAB-.git}$

### Program 1

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

3/3/25	Lab-1.	1.1
7	Lab. 1 Page No.:	. Which column to the dataset had mising
		Value 9 How did you handles them?
	Write puttoni and	
	"housing csv"	Propped the missing values with the
(1)	To load cay in the	help of drop ma dropnox since
ù	To load cer file into the data frame.	the number of some are beginthe nuising value is less compatibility
	THE DYNAMING SI ALL COLUMN IN A	missing value is less compatible
	satisfied information of all	2
in	To display the count of unique labels 2.	What were the categorial date columns did
10)	To display the count of unique labels	and the fire the dancer
	The Isoximity tolumin	you encode them?
21		Fro a waller
	missing value count	diabetes . cev: ["Gender", "class"]  adult . cev: ["wortedace", "education", max stal
_	greater than zero.	adult-cev " " "relationship", " gender"
_		occupation telestores
	A George Control of the Control of t	adult cev: [ "voortedace", "education," nav sail "occupation", "relationship", "gender", "noting country", "income".
(1),	(pd. df > pd. read-czv ("housing.czv")	and we used label encounter to encode
	The state of the s	lind we used to
(ù)	print (dt)	it.
		between Min - play
(20)	print (df-describe ()).	Chaling and standardization & when
	2,	cooling and standardization other?
(iv).	print (of ["Ocean Proximity"], value_count(1)	WORLD GOLD
	( ) value_com()	A salls All
(v)	Missing values all builts and	Both Min Mar Scaling and standardisation
1	missing-value = of inul().sum() >0	are feature scaling technique.
	missing - column = of missing values missing mis	1 × × / 1 × × × /
	print (missing column).	X = X - Xmin x = X - 4
		Max Ahin
-	The state of the s	
_		
	we use nun max when we do not h	ave
	to the transfer of the transfe	(84)
	eignificant outlier.	1130
	and we use standardization when	WE
	have outliers	
	Marc out area	
	1	
	A Comment of the Comm	
	capte and been a break out my steel	
_	Section of the pass of the passes of the pas	II ees

```
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

10/03/25	LAB-2 Spale		S Date :
	Demonstrate the steps to build a machine - learning model that predicts the median	3	Demonstrate process of creating set to differences blus random & stratified feet
	housing price using the California housing price dataset.	Teans	Random Stratified.  Splits data randomly Both are represented that training 8 lump of the events distribution of
1.	Perform the describe and info steps:		a teature
	import pandas as pd housing: pd. read-cav ("content/drive/"). print (bousing. Into()) print (housing. describe ())		May not present the Preserve the destribut distribution of East of East features feature in training across training to testing set. To testing
	Output: column = 9 nows = 20640.		hist geographical feature.
2.	Plot histogram for each feature	-3-47	This graph visualizes housing prices in relation to their geographic location, with the color representing medium house value and sire worresults.
	the histogram for median income & hours - median age can give us included into the distribution of these features. For example, median income might	d'a s	honse value and size represents
	that most households have lower	5	which feature correlates to maximm.
	might show how the ages of houses are distributed.	_/	highest correlation with the median

	Page No.
B	list the feature that could improve correlation.
	By creating nome per household & bedrooms per nom.
1)	The median house value is improved
7	total bedrooms list the feature that leader data total cleaned & demonstrate cleaning process.
	total bedrooms.
<b>(2)</b>	Categorial data to numerical data.
2 33.50	Yes, ocean proximity is categorial feature and method used to convert is One flot facoder()
2	Importante Scaling feature:
	ensure all feature contribute equally to result. Technique used Standard Scaler &
<b>Q</b>	The state of the s

```
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)
```

Output:

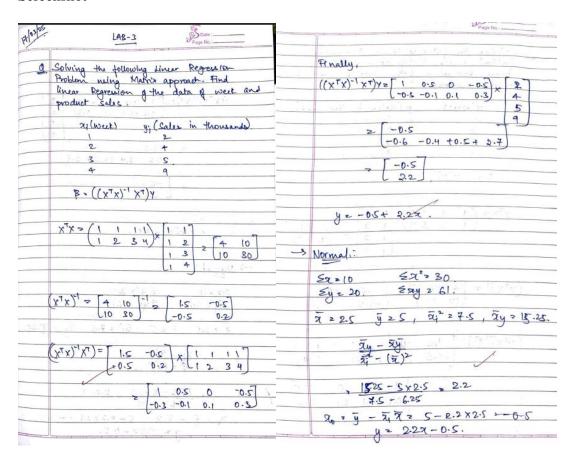
```
60
                                                             No Pation
                                                             Gender
Missing values in Diabetes dataset after removal:
                                                                           56
                                                             AGE
                                                                           39
ID
             0
No_Pation
            0
                                                             Urea
                                                                           58
Gender
                                                             Cr
                                                                           48
AGE
            0
                                                            HbA1c
                                                                           42
Urea
            0
                                                             Chol
                                                                           47
Cr
                                                             TG
                                                                           49
HbA1c
            0
                                                            HDL
                                                                           54
Chol
            0
                                                             LDL
                                                                           40
TG
            0
HDL
            0
                                                             VLDL
                                                                           46
LDL
            0
                                                             BMI
                                                                           51
VLDL
            0
                                                             CLASS
                                                                           48
BMI
            0
                                                            dtype: int64
CLASS
            0
dtype: int64
                                                            Missing values in Adult dataset before removal:
                                                                                  2304
Missing values in Adult dataset after removal:
                                                             age
                                                             workclass
                                                                                 2324
                  0
workclass
                  0
                                                             fnlwgt
                                                                                 2325
fnlwgt
                  0
                                                             education
                                                                                 2403
education
                  0
                                                             educational-num
                                                                                 2392
educational-num
                  0
                                                                                 2446
                                                             marital-status
marital-status
                  0
                                                             occupation
                                                                                 2480
occupation
                  0
                                                             relationship
                                                                                 2402
relationship
                  0
                                                             race
                                                                                 2379
race
                  0
                                                             gender
                                                                                 2463
gender
                  0
                                                             capital-gain
                                                                                 2369
capital-gain
                  0
capital-loss
                  0
                                                             capital-loss
                                                                                 2376
hours-per-week
                  0
                                                             hours-per-week
                                                                                 2367
native-country
                  0
                                                             native-country
                                                                                 2382
                  0
income
                                                             income
                                                                                 2324
dtype: int64
                                                             dtype: int64
```

Missing values in Diabetes dataset before removal:

13

```
Categorical Columns in Diabetes Dataset: Index(['Gender', 'CLASS'], dtype='object')
Categorical Columns in Adult Dataset: Index(['workclass', 'education', 'marital-status', 'occupation', 'relationship', 'race', 'gender', 'native-country', 'income'],
        dtype='object')
  Encoded Diabetes Dataset:
        ID No_Pation Gender AGE Urea Cr HbA1c Chol TG HDL LDL \
  0 502.0
             17975.0
                           0 50.0 4.7 46.0
                                                 4.9 4.2 0.9 2.4 1.4
  1 735.0
                           1 26.0 4.5 62.0
                                                  4.9 3.7 NaN NaN 2.1
             47975.0
  2 420.0
                           0 50.0 4.7 46.0
                                                  4.9 4.2 0.9 2.4 1.4
                         0 50.0 4.7 46.0 4.9 4.2 0.9 2.4 1.4
  3 680.0 87656.0
  4 504.0 34223.0
                           1 33.0 NaN 46.0 4.9 4.9 1.0 0.8 2.0
     VLDL BMI CLASS
    0.5 24.0
  0
                    0
  1
     0.6 23.0
                     0
  2
     0.5 24.0
                     0
     0.5 24.0
                     0
  4 NaN 21.0
  Encoded Adult Dataset:
                       fnlwgt education educational-num marital-status \
      age workclass
  0 25.0
                 4 226802.0
                                                    7.0
                                      1
  1 38.0
                   4 89814.0
                                                      9.0
                                      11
                 2 336951.0
  2 28.0
                                       7
                                                     12.0
                  4 160323.0
  3 44.0
                                      15
                                                     10.0
                                                                        2
  4 18.0
                  0 103497.0
                                      15
                                                     10.0
     occupation relationship race gender capital-gain capital-loss \
  0
                                 2
                                        1
                                                    0.0
                                                                   0.0
                            0
                                                     0.0
             11
  3
             7
                                                  7688.0
                                         1
  4
             0
                           3
                                 4
                                         0
                                                    0.0
                                                                   0.0
     hours-per-week native-country income
  0
              40.0
                                         0
                                39
              50.0
                                 39
  1
                                         0
  2
              40.0
                                 39
                                         1
  3
               40.0
                                 39
                                         1
  4
              30.0
                                 39
                                         0
   Feature MinMax_Scaled Standardized
                0.000000
0
       10
                             -0.502219
                0.010101
                             -0.474574
1
       20
                0.020202
                             -0.446929
2
       30
3
       40
                0.030303
                             -0.419284
4
       50
                0.040404
                             -0.391639
5
      1000
                1.000000
                             2.234643
```

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



The state of the s	=	S Caste:
U Did you potorm any data processing steps?	-	Page (to)
with the estimating value by filling them	T.,,	Did you real the feature ? If yes why?
label encoding to categorial (like: 2 4tht) ca and escaled numerical features for 1000-companies cer to normalize the data.		Yes, because of R&D Spend, Administration, & Marketing Spend have
2) Did you visualize the regression line for	l' de	(using Standard Scater (1) was
Yes How has	-luh	applied to improve model performance.
thouse a strong line was plotted. The plot thouse a strong linear relationship between that as the fear increases, per capita income also trees		and an area of the substances and the second
3 Predicted salary for (10 years experience, 10 feat score, 10 interview score) 9 P		Construction of the second
Sy & sore, 10 interview score) ) 7	to a	of the comment with a
The predicted salary is printed in the script and depends on the trained models coefficient		The property will be to take
1000 companies caregonal variables for		-33571 0. fc 1-10.577 Affice
Yes, the "State" column was encoded using dabdencoder().	- 52	e i made as many the many the second of
The state of the s	-	the of believe to queles briefland all

```
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()
```

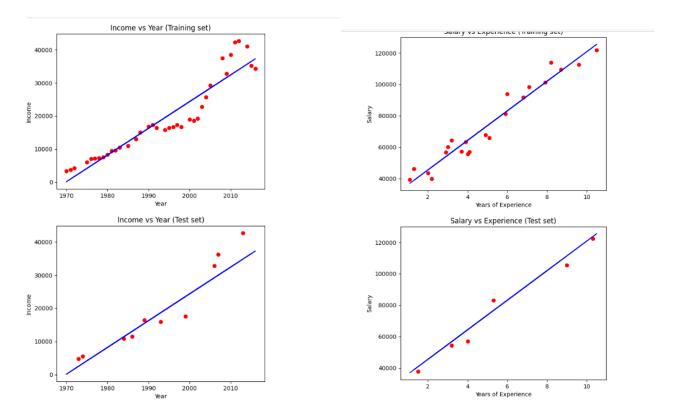
```
plt.grid(True)
plt.show()
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.impute import SimpleImputer
# Load the dataset
df = pd.read_csv('/content/salary.csv')
# Prepare the data
X = df.iloc[:, :-1].values # Features (years of experience)
y = df.iloc[:, 1].values # 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}")
```

### Output:



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

A 103/25 Lab-4 Spain:	S Date
Page No.:	P(1) 2 . 1
Logistic Penguin - (1)	P(1) 2 e1 0.0900 .
Logistic Regression:	
(1). Shedent - passed or tail based on the shidy	P(0) 2 e 0.2448
1. Student - passed or fail based on the study Intercept 9:2-5 isofficient? 2,20.8	P(0) 2 e° 0.2448
	an equal to explain.
(6) P(Z) = 1 Z = a, + al. x 1+e-2 Z = 15 + 0.8 x.	For HR_comma_cep.cov.
	0.1 Which variable had clear and direct
touch have used but all you drawed ()	Impact on employer retention? why?
(B) Probability that shident who shidies for	Sol satisfaction level, time spent company, number project, average monthly houre
P(7) = 1 + esse - 1.4 e of the time	Reason: They showed strong correlation with retention based on EDA.
@ Determine predicted class (P/F) for stident based on threehold o.c	e Q-2 Accuracy of together tegretation model "> Goo
(Apper de la programa	Accuracy (eg n 79%)
As 0.6456 > 0.5 this statement will Pace	Accuracy (eg n 79%)
i d'account la	Ye, it has good accuracy it is
Q2. Consider 22 (2110) iter 3 charge took	may be possible.
syllagues function to tind probability	may be possible william
value of 3 classes.	1 / 20 1 200
7	
P(z) z ez 2 0.6652	
D(2) 2 02 2 12 (c) 2 1862.	
60 + 01 + 6 2 5 0.8035	

```
For 200 dataset
I Did you perform data procusing steps !
   propped animal home as its not needed for dazzification.
i) Where there any muchy values ?
    No missing values defeated
(ii) what does the confusion matrix tell you
    about the performance
    Shows class- wice patermance
   It shows how many instance of each class were correctly or incorrectly daniped It helps ares accuracy and identity
              miscalculation
in which class type were most frequentry
    Class types with similare teatures (eg mamne
    & humans) were most frequently misolossis
          happened due to overlapping
       features and
                          menticient
```

```
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

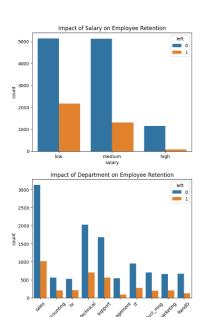
```
y_pred = model.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
```

print("Accuracy of the model:", accuracy)

### Output:

	last_evaluation num		age_montly		\
0.38	0.53	2		157	
0.80	0.86	5		262	
0.11	0.88	7		272	
0.72	0.87	5		223	
0.37	0.52	2		159	
time spend company	Work accident left	promotion last 5	vears Depa	rtment	\
3	0 1		9	sales	•
6	0 1		0	sales	
4	0 1		9	sales	
5	0 1		0	sales	
3	0 1		0	sales	
salary					
low					
medium					
medium					
low					
low					
lass 'pandas.core.fr	ame.DataFrame'>				
ngeIndex: 14999 entr					
ta columns (total 10					
Column	Non-Null Count	Dtype			
satisfaction_leve	1 14999 non-null	float64			
last evaluation	14999 non-null	float64			
number project	14999 non-null	int64			
average montly ho	urs 14999 non-null	int64			
time_spend_compan					
Work accident	14999 non-null	int64			
left	14999 non-null	int64			
promotion last 5y	ears 14999 non-null	int64			
Department	14999 non-null				
salary	14999 non-null	object			
ypes: float64(2), in	t64(6), object(2)	•			
mory usage: 1.1+ MB					
ne					
	vel last_evaluation		\		
unt 14999.000		14999.000000			
an 0.612		3.803054			
d 0.248		1.232592			
n 0.098		2.000000			
% 0.448		3.000000			
% 0.648		4.000000			
% 0.828		5.000000			
x 1.000	1.000000	7.000000			
average_montly	hours time_spend_com	pany Work_accide	nt	left	\
unt 14999.0		0000 14999.0000		000000	
an 201.0	50337 3.49	0.1446	10 0.	238083	
		0.3517		425924	
		0.0000		000000	
		0.0000		000000	
		0.0000		000000	
		99999 9,99999		000000	
	100000 10.00			000000	
	20100	210000			
promotion last	5vears				



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

#### Screenshot

	Page No.
	Decision Free: . to the not set
6 - 0	Entropy (FT) = E. P. log Pi
1.4	Entropy (T, A) = & lail is entropy (A)
	2G(A) + (T) - E(T, A)
	Salfin A-Judy with me
	5 S S S S S S S S S S S S S S S S S S S
	statement of the steel could
· .ch	of property property and the fit
g. Har	Is the country of the first
6	Leaves to the second state of the second state of the second seco
	- # # # # # # # # # # # # # # # # # # #
5.675.1 ci )	A section of a section of the sectio
	all and the second of
	find the said

#### 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()
Output:
```

Iris Dataset Re Accuracy: 1.00 Confusion Matri [[10 0 0] [ 0 9 0] [ 0 0 11]] Classification	x:				Drug Dataset Ke Accuracy: 1.00 Confusion Matri [[6 0 0 0] [0 0 5 0] [0 0 0 1] [0 0 0 1] Classification	ix: 0] 0] 0] 0]	recall	f1-score	support
C1833111C8C1011	precision	recall	f1-score	support	0	1.00	1.00	1.00	6
Iris-setosa	1.00	1.00	1.00	10	1	1.00	1.00	1.00	3
Iris-versicolor		1.00	1.00	9	2	1.00	1.00	1.00	5
Iris-versicolor		1.00	1.00	11	3	1.00	1.00	1.00	11
iris-virginica	1.00	1.00	1.00	11	4	1.00	1.00	1.00	15
accuracy			1.00	30	200112201			1.00	40
macro avg	1.00	1.00	1.00	30	accuracy				
weighted avg		1.00	1.00	30	macro avg	1.00	1.00	1.00	40
3					weighted avg	1.00	1.00	1.00	40

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

#### Screenshot

K- Nearest Neighbour Fronts	and the state of the page No.
Consider following dataset, k = 3 and test data (x, 30, 100) as (Perton, Age, Salary) softer, using the know descripter and product targets with uson in 52.21  Person hage solver wing the form of 100 in 52.21  Person hage solver with uson in 65.27  D 411 60 y 40.44  E 12 + ortotop in y 21.04  Nearest wighbours = 510, D  Nearest wighbours = 510, D  Nearest wighbours = 510, D  Coored = E ((y, 10x)) and the following the	target - 7  to tor vie dataset the k value? Demonstration of those validation to used to find the best t value depending on occuracy or number of new chosen.  To the chosen in the purpose of teature scaling?  What is the purpose of teature scaling?  There is calling it used in KNN to have the same scale because if we have the same scale because if we have the same scale because if we have the same scale get more weightige so larger scale get more weightige so larger scale get more weightige so larger scale get more weightige so act supressed and wont have an
Concila Nome of (N, Y) & (N, N)  + (N, Y) & + 0 & 1	caling this the frequency distributions  caling this the frequency distributions  all features were charted the  in an having normal distributions were  ecoled insing thandard scaler and he  rest ming minmax caler.
1	

#### 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.")
  exit()
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))
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_{test} = scaler.transform(X_{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
# 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)
# Plot
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)
Output:
```

```
IRIS Dataset - Actual vs Predicted:
```

	Actual	Predicted
0	versicolor	versicolor
1	setosa	setosa
2	virginica	virginica
3	versicolor	versicolor
4	versicolor	versicolor
5	setosa	setosa
6	versicolor	versicolor
7	virginica	virginica
8	versicolor	versicolor
9	versicolor	versicolor
10	virginica	virginica
11	setosa	setosa
12	setosa	setosa
13	setosa	setosa
14	setosa	setosa
15	versicolor	versicolor
16	virginica	virginica
17	versicolor	versicolor
18	versicolor	versicolor
19	virginica	virginica
20	setosa	setosa
21	virginica	virginica
22	setosa	setosa
23	virginica	virginica
24	virginica	virginica
25	virginica	virginica
26	virginica	virginica
27	virginica	virginica
28	setosa	setosa
29	setosa	setosa

#### Accuracy Score: 1.0

Confusion Matrix: [[10 0 0] [ 0 9 0] [ 0 0 11]]

#### Classification Report:

	precision	recall	+1-score	suppor
setosa	1.00	1.00	1.00	10
versicolor	1.00	1.00	1.00	9
virginica	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

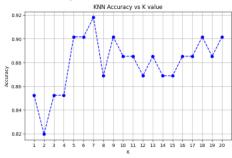
DIABETES Dataset - Actual vs Predicted:

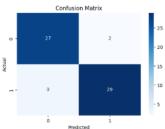
	Actual	Predicted
0	9	9
1	0	9
2	0	9
3	0	9
4	9	1
149	1	1
150	0	9
151	0	9
152	1	9
153	9	9
[154	rows x	2 columns]

Accuracy Score: 0.6948051948051948

Confusion Matrix: [[79 20] [27 28]]

Best K value: 7 with Accuracy: 0.9180



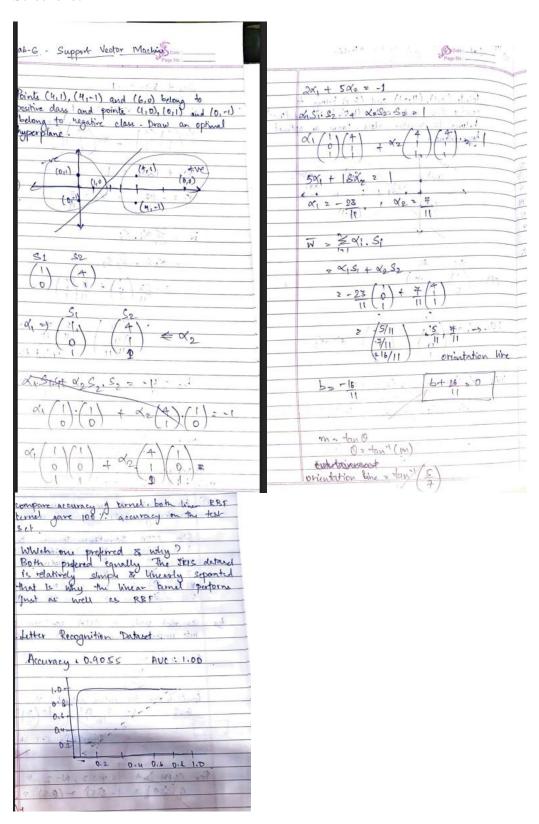


Classification Report:

	precision recall f		f1-score	support	
8	8.98	8.93	8.92	29	
1	0.94	8.91	8.92	32	
accuracy			8.92	61	
macro avg	0.92	0.92	8.92	61	
weighted avg	0.92	8.92	8.92	61	

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

#### Screenshot



```
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')
```

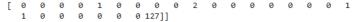
```
X = data.drop('species', axis=1)
y = data['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
svm_rbf = SVC(kernel='rbf')
svm_rbf.fit(X_train, y_train)
y_pred_rbf = svm_rbf.predict(X_test)
accuracy_rbf = accuracy_score(y_test, y_pred_rbf)
cm_rbf = 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')
```

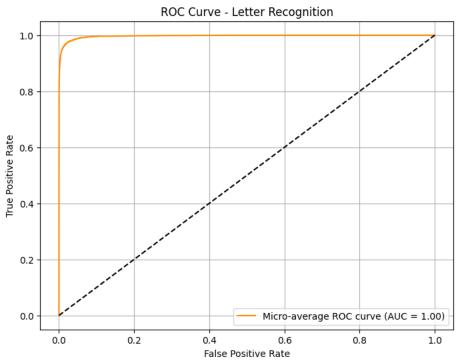
```
svm_linear.fit(X_train, y_train)
y_pred_linear = svm_linear.predict(X_test)
accuracy_linear = accuracy_score(y_test, y_pred_linear)
cm_linear = confusion_matrix(y_test, y_pred_linear)
print("\nSVM with Linear Kernel:")
print("Accuracy:", accuracy_linear)
print("Confusion Matrix:\n", cm_linear)
plt.figure(figsize=(6, 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']}")
Output:
                             ---- Letter Recognition Dataset ----
                             Accuracy: 0.95025
                             Confusion Matrix:
                             [[148
                                   0
                                       0
                                        0
                                    0
                                           0
                                               0
                                                  0
                                                      0
                                                          0]
                                1 147
                                       0
                                               0
                                                  0
                                                      0
                                       0
                                               0
                                           0
                                                  0
                                                      0
                                                          0]
                                    0 126
                                           0
                                               2
                                                   0
                                                      2
                                           0
                                               0
                                                   0
                                        0 153
                                               0
                                                  0
                                                      0
                                               0
                                                          1]
                                               1
                                                 134
                                                          0]
                                               0
                                                  0 153
                                       1
                                           3
                                                          0
                                               0
                                                          0]
                                                  0
                                                      0
                                               0
                                                             0
                                        0
                                                  0
                                                      1 118
                                                                 0
                                               0
                                                   1
                                                      1
                                                          01
                                               0
                                                   2
                                                          0 136
                                                                     0
                                               0
                                                      0
                                                             2 143
                                                                     0
                                                   0
                                                      0
                                                          0]
                                               0
                                                          0]
                                               0
                                                   0
                                                          11
  ---- IRIS Dataset ----
                                               0
                                                   0
                                                      0
                                                                        0 166
                                                                                   0
                                           0
                                               0
                                                   0
                                                      a
                                                          01
                                0
                                               0
                                                      0
                                                   0
                                                                            0 140
  Accuracy: 1.0
                                               1
                                                   0
                                                      0
                                                          01
  Confusion Matrix:
                                0
                                        0
                                               0
                                                   0
                                                      1
                                                                                0 136
  [ 0 9 0]
[ 0 0 11]]
                                               2
                                                  0
                                                          0]
                              [
                                0
                                               2
                                                  11
                                           0
                                               0
                                                  0
                                                      0
                                                          0]
                              [
  RBF Kernel:
                                                          0]
  Accuracy: 1.0
  Confusion Matrix:
                                                          0]
  [[10 0 0]
                                0
                                                      0
                                    2
                                       0
                                           0
                                               1
                                                  0
                                                          0
                                                             0
                                                                     0
                                                                            0
                                                                                   0
                                                                                       0
  [0 9 0]
                                       0
                                               0
                                                      0
                                           0
                                                  0
                                                          1]
  [0 0 11]]
                                           0
                                                      1
                                                             0
```





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

# Screenshot

oclospis	Lab. 7 South	S Care
0=1	Page No.	for CGPA <9 -> 1 sample
		Y = 1, N = 0.
	Random Forest	Gini (29) 70.
	Ensemble	
		Weighted Ginicapo 2 4 0.5+ 1 x 0= 0.4
O	For sample SI draw decision tree	CGPN S'
trile !	considering CAPA as not node	
	10	Internation Gain (Gini needletis)
	SNo CGPA Interactioners Common knowled Ja	= 0.48 - 0.4 2 0.08
	1 A Good Good Y	- 0.73 - 0.74 - 0.02
+7.1 (lo	21 K9 N Mederation Good 4	- Court ocon so that the
Long	3 . Ag N . My My	Split CGPA >9 further to get.
	WE THE ME THE NEW YORK NOW THE THE N	split an interaction y - 100 vcs-2
	5 29 12 Y 11 M Good y	N - N - 2
		fin > 2 x 0 + 2 x 0 00.
A.,,	a noot node is CGPA we can seperate	fini = = x 0 + 2 x 0 = 0.
	into unique togroups tax > 29 18:129	
		(gini reduction) 9 a 0.5. 0 2 0.5.
19	For cigPAI≥9 12NO 1 12NO	10.4 (2004) 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
	Y:2	CG PA Lond of Land
	N:2	C9
	Constitution of the subject : 4-3, N-2	2
	G. I 1- 5 D2 2 1- (3)2 (2)2	
	101 (3)	yes internet No Yes.
	Civi(D) = 1-0.36-0.16	The contract of the contract o
	2 0.48	
	9.5 . 10 . 20 . 10 . 10	
đ	or (4PA > 9 4:2, N:2	Yes
1	Or (414)	1100
-	9(59) 2 1- (0.5) ~ (0.5) 2 20.5	
-		
-		

Givi = $1 \cdot \left(\frac{1}{5}\right)^2 - \left(\frac{1}{5}\right)^2 = 1 - 0.64 - 0.04$ $= 0.32$	
· Chris 3 x 0 + 3 x 0 400 > 0.2664	Fuge tto:
Sqini = 0.32 - 0.2664 = 0.053	RF M-estimate mean accuracy.
	The state of the s
N N N N N N N N N N N N N N N N N N N	0.7166-1
growing the "no" beach with other	2 100 0.9667
features.	100
1000	130 0.1007
veighted gini> 1x0+ 2x0.50 > 0.33)	\$ 200 0.9667.
A Giniz O.III	3 6 9
	Accuracy = 1.0
Practiced transledge: Locignited gini = 0. Agini = 0.444 (best).	Best number of tree = 20
Agini? a.ynu (best).	confrain matrix - 10 10,0
(Interactioners)	0 9 0
	(0 0 ").
Yes No:	MATERIAL TO THE THE
	V. The state of th
"Yes" [Practical knowledge]	A 21 105.
Good Averego	took at its
Yes No	

#### 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()
Output:
  Default n_estimators=10 → Mean CV accuracy: 0.9667
  Random Forest Tuning Results n_estimators mean_accuracy
            10
                    0.966667
                    0.966667
            50
            100
                    0.966667
            150
                    0.966667
            200
                    0.966667
  Best number of trees: 10 → Mean CV accuracy: 0.9667
```

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

#### Screenshot

105/25	dab		Page No.							
	Boosti	ng Grocub	1 10							
	CRPA' Intrac	Practi	Com	Jol					3. 6	Some _ 20 2gt .
	>-9 Y	TYRE!	G	y						rego no.
	29 N	9	M	Y		· · · · Lin	Herael .		1	France . The
	>9 N	A	M	N	wit is	15 .	Mil-	Act		"reves"
	<9 N	A	G	N	. 1		V (F 3.)	1.4.		01.2501
	≥9 Y	G	M	Y			Y	V	-1 -1	
	>9 Y	- 1	M	y			n	N	2011/4/4	0.2501
		9	•				4	Y		1.1249
	Initial weight	1. Va.	Jan Street				Y	·		0.1249
	Just weight	6	D	re Ac			Sact .	ed took		10 1019
-	Table 4			4	21.4487.2	4	18,51			5.11 Jr. 11.61
	= 1.1.1.V	14(11)	N	V	-	Adab	est all	n-esh.		1 1
	5, ti; (dj)	·Wilay			10	1	1. 1.	10		Light -
	U		, Y	J N	- 15	2	41.4	50 -1		110122037
	271	0.33		) N	- 10	2	13.8	100 14.		1 0 1 6 3 0 0
	6			1 7	0.7		0.0	12010		111 0,83050
	√~ j ln	(1 - Ecupa	) >	Y		C	3. 6			16.01.1322
	2	ECHPO				->		800.		1:0:10 326.
		cara			- 6.)		40.1	16		12,2,41
	Keter -	In/1-	0.33		0			92.	11	(2.01,2.2)
_	CAPA	> 1	0.33			cont	usion			
			0.33			1		7028	1386 -	
		-1 .	00		/			1243	1412	tions engly
	× Cler	, g. 0.	397			Qu)	1	1244		
					55.	- 1	34	41944	2	9-1 - 10
					-		-		3	3.10
	_ \	1 x e -0.0.344	. 1 . 2	たんのまけ						
	7 6	10	6							
		> .				245	17 1 2	1 221	2.01.2	Stara
		0.942	1		_		15		1	
	new 2	1210			F5 X	. 27.10				

#### 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()
```

## Output:

Best performance: 0.8326 accuracy using 200 trees

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

#### Screenshot

	Page No.:
	n tela
-	Heration 2 (1.82, 2:32) (4.12, C.27) Assign
Jab-9 Sade	Record Close to C1 Close to Ce
Prago No.:	(1.1) -1.57 6.37 4
	· (1.572) . ONT +273.19
a Compute two clusters wery kniegne	(8,4) 2.04 1.77 C2
algorithm for clustering whose cluster centers are (1,1) & (5,7) execute	(5,2)
for 2 Heration.	(2.5,5) +1 3.15 + 11 0.72 C2
TOV 2 HONTON	(4.5,5) 3.78 0.53 C2
> Theration 1:	1-11.
S CHETOLOGY Z.	7 1 2
Cale fuclidean doct to centroide:	trental entre orthobrant
(1,1) (S,4) Acsignment	New Chreter are
Record close to () close to (2)	C1 = & R1, R28 , C2 = & R8, R4, R5, R6
(1.1) 0 7.21 4	1 6, 34 6131.4131.8
(1.5.2) 1.12 6.12 4	New Centroide ( > = + = 11) = ,8
(3,4) 3.61 3.61 C	New Could be 12 2 1 1 1
92 0.0 19.7 (4,2)	
(2.5,5) 4.12 2.5 Co	C1 = 2.5 3 (0 = 19.5 25.5 2-1 2 2 2 1 5 5
(4.5.5) 5.31 2.06 Ce	8-1 2-9 2-2 2-1 5 5
(2.5, 4.5) 4.30 2.92 62	1 3-11 3-3 9-11 3-11
anti-day	
- n.281 > 20 % of	For Inc dataset
New centraids:	
	( ) 1 2.2 J.H . 39 )
0.5 1+1.5+3 . 4+2+1 1.83, 2.82	The elbow plot (herora work) shows a
(1 = 1+1.5+3 , 4+2+1 , 1.83, 2.3)	sharp elbow at kis, indicating that the churter is the nomel choice for.
	A. I I I I I I
A ALBERT DE MITTERINE	the culeter to the named choice for.
C2 > 5+3.5+4.5+3.5, 7+5+ C+4.5	potal length and with o the love.
7 4	1 1.1.
= 4.12, 5.37	1/1/2010/10/10/1
	1 1 1 1 10 10 1 10 1 10 1 10 1 10 10 10

#### 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_{\text{test\_scaled}} = \text{scaler.transform}(X_{\text{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)
```

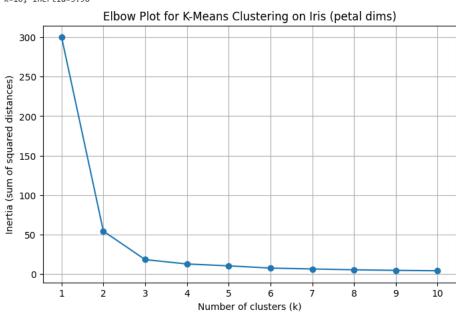
```
# Predict on test data
predictions = kmeans.predict(X_test_scaled)
# Note: There's no ground truth labels, but for demonstration,
# we can try assigning true clusters (via KMeans on full data)
# and see if predicted clusters align
# Fit on full data to assign pseudo-labels
full_kmeans = KMeans(n_clusters=optimal_k, 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()
```

# Output:

k=1, inertia=300.00 k=2, inertia=54.15 k=3, inertia=18.05 k=4, inertia=12.52 k=5, inertia=10.14 k=6, inertia=7.31 k=7, inertia=6.19 k=8, inertia=5.16 k=9, inertia=4.41 k=10, inertia=3.90



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

# Screenshot

plostes	S Date:	Scale:
	dab 11	Page No.
0.4	e contract (september )	
	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	2 1 -4 0 61 -4 - 2.5 2.5 -4.5 -1-3.5 . C.S.) 0 -4.5
10	82 g (1d)	3 (2.5 -4.5 -1-2.5 . 5.5) 0 -4.5
Q.	Reduce dimination from 2 to 1.	2.5-
20 20 20 20 20 20 20 20 20 20 20 20 20 2	8G. (U) ( (#.1t)	1.1-1,1965
+1	XI. 4 8 -18 7 (0.1)	
45	X== 11 + =15 14 (= 20)	2 -38 69
50	END SEE (2.5M)	3 (-33 / 69 /
2	F1. (F.C (3.4,2.9)	
	Standadize the dataset	C2 [14 -11 prilament]
	then stanks are	(-11 23)
19 34	10 - 1347 - 18 8 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	Let ((C - AI) = 0.50
	4	Let (C - AI) = 0 . TV
	82 = (1+4+5+ 14) 0 18 35	,
		14-2-20-11 - 0.
2.2	XC = 4-c -8-c 13-c 7-8	- 11 05 3 23-A
× 1	XC = 4-6 -80 12-8 7-8	
	11-85 4-8 5-8 14-8	12 so 2 + 37. + 201 7 0 x 1/2)
-		
	Xc > -4 Ounters aid - 113?	1 1 1 2 30, 2849, P- 1, y y 23
	25 -4.5 -3.5 6.5	2083.07 2.8- 0
4. 0	work (38th wheed ) tells rundle soll	consider larger 1
1.2.41	Covariante madrix to made quell	(0, ) 1-
101	the streets is they is a lost	14-20.3849 -11 7 7 7
. 24	de ( n. ) de x then of sult	-11 202, 4 1-23 -30, 3841 1 9 0
		1,456.5
	3/25 45-35 155	-16. seyon 202114 . [6]
	2/502 1/12/-3/2 /2.2/	-16. seyga 20-11g 0 0 0 -11x 115+7-5849y 0
		The string of th

Prage No :	Spage to:
-16.3849x -11y = 0	Plo
taking y, 1	Acceuracy before PCA
(y) <sup>2</sup> (-0.6713)	logistic regression: 0.9016 sym: 0.8525. Random forest: 0.8361
Normalize 11- 11 . 9	
7 2 x y = V x 2 + 1 2 - 1 x 2 + 1	Accuracy After PCA logistic: 0.8689 SVM: 0.8689.
C = [-0.5574-1" 1 0:8803:1	Random forcet: 0.88.
Calculate principle corriponent.	
Z-XeC1 = -4 = 2.5 [-0.55.74]	
C +3.5 mm = 1 = 1 = 1 = 1 = 1 = 1 = 1 = 1 = 1	
M 100 2 2 1 4 30 5 11.	
-3.736' -51698	
(233.121)	

#### Code

<a

href="https://colab.research.google.com/github/mukund166/ML\_Lab\_1BM22CS166/blob/main/1BM22CS166\_Lab\_10\_PCA.ipynb" target="\_parent"><img src="https://colab.research.google.com/assets/colab-badge.svg" alt="Open In Colab"/></a>

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.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])
# 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)
```

from sklearn.ensemble import RandomForestClassifier

```
], remainder="passthrough")),
  ("scaler", StandardScaler())
])
# 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}")
Output:
 === Accuracies BEFORE PCA ===
             : 0.8750
 LogisticRegression: 0.8533
             : 0.8641
 PCA retained 13 components, explained variance = 0.9719
 === Accuracies AFTER PCA ===
 LogisticRegression: 0.8533
 RandomForest
```