DRONACHARYA
College of Engineering

LABORATORY MANUAL

B.Tech. Semester- VI

STATISTICAL MACHINE LEARNING LAB
Subject code: LC-AI-344G

Prepared by:	**Checked by:**	**Approved by:**
Prof. Naveen	Dr Ritu Pahwa	Name : Prof. (Dr.) Isha Malhotra
Sign.: ……	**Sign.: ……**	**Sign.: ……**

DEPARTMENT OF CSE(AI&ML)
DRONACHARYA COLLEGE OF ENGINEERING
KHENTAWAS, FARRUKH NAGAR, GURUGRAM (HARYANA)

Table of Contents

Vision and Mission of the Institute

Vision:

"To impart Quality Education, to give an enviable growth to seekers of learning, to groom them as World Class Engineers and managers competent to match the expending expectations of the Corporate World has been ever enlarging vision extending to new horizons of Dronacharya College of Engineering"

Mission:

M1: To prepare students for full and ethical participation in a diverse society and encourage lifelong learning by following the principle of 'Shiksha evam Sahayata' i.e., Education & Help.

M2: To impart high-quality education, knowledge and technology through rigorous academic programs, cutting-edge research, & Industry collaborations, with a focus on producing engineers& managers who are socially responsible, globally aware, & equipped to address complex challenges.

M3: Educate students in the best practices of the field as well as integrate the latest research into the academics.

M4: Provide quality learning experiences through effective classroom practices, innovative teaching practices and opportunities for meaningful interactions between students and faculty.

M5: To devise and implement programmes of education in technology that are relevant to the changing needs of society, in terms of breadth of diversity and depth of specialization.

Vision and Mission of the Department

Vision:

To cultivate skills and make proficient engineers cum trainers in the domain of Artificial Intelligence & Machine Learning for exceptional contributions to the society.

Mission:

M1: To impart intense training and learning to generate knowledge through the state-of-the-art concepts and technologies in Artificial Intelligence and Machine Learning.

M2: To establish centres of excellence by collaborating with the leading industries to exhilarate innovative research and development in AIML and its allied technology.

M3: To inculcate regenerative self-learning abilities, team spirit, and professional ethics among the students for noble cause.

Programme Educational Objectives (PEOs)

PEO1- ANALYTICAL SKILLS:

Using a solid foundation in mathematical, scientific, engineering, and current computing principles, formulate, analyse, and resolve engineering issues in real-world domains.

PEO2- TECHNICAL SKILLS:

Apply artificial intelligence theory and concepts to analyse the requirements, realise technical specifications, and design engineering solutions.

PEO3- SOFT SKILLS:

Through inter-disciplinary projects and a variety of professional activities, demonstrate technical proficiency, AI competency, and foster collaborative learning and a sense of teamwork.

PEO4- PROFESSIONAL ETHICS:

Excel as socially responsible engineers or entrepreneurs with high moral and ethical standards, competence, and soft skills that will enable them to contribute to societal demands and achieve sustainable advancement in emerging computer technologies.

PROGRAM OUTCOMES (POs)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

P11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

P12: Life-long learning: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Fundamentals:
Apply the knowledge gained pertaining to data storage, data analytics and AI concepts to solve real-world business problems.

PSO2: Applications:
Ability to evaluate and apply knowledge of data engineering, artificial intelligence, machine learning, and human cognition to real-world issues in order to solve potential challenges.

PSO3: Innovation:
Ability to acquire computational knowledge and project development abilities using novel tools and methodologies to tackle challenges in the fields related to Deep Learning, Machine learning, Artificial Intelligence.

PSO4: Implications:
Capacity to direct a team or firm that develops products and to use the knowledge learned to recognise actual research issues

University Syllabus

Course code	LC-AI-344G				
Category	Professional core courses				
Course title	Statistical Machine Learning Lab				
Scheme and Credits	L	T	P	Credits	Semester = 6
	0	0	3	1.5	
Classwork	25 Marks				
Exam	25 Marks				
Total	50 Marks				
Duration of Exam	03 Hours				

NOTE:

1. Minimum 15 Lab programs/activities can be designed and developed by the subject faculty using Python. Python Library/suitable Open-Source tools/ software.
2. Lab activities will be carried out from the offered course contents of Statistical Machine Learning in the semester.

In this course, various experiments will be performed, covering Statistical Machine Learning techniques. Experiments covering pre-processing of data, various classifiers such as Bayesian, Decision Trees, Support Vector Machines, k-nearest neighbour; Regression Models, and data sets will be described in the laboratory manual. Measures of classification precision, enhancement of classifier efficiency by the assembly, boosting, PCA, SVD on handwritten digits using Scikit-learn etc.

Course Outcomes (COs)

Upon successful completion of the course, the students will be able to:

CO1: Implement python programs and machine learning algorithms.
CO2: Analyse the trends in datasets using descriptive statistics.
CO3: Apply descriptive and predictive modelling.
CO4: Compare and contrast machine learning algorithms for a given problem. (Describe datasets using descriptive statistics.
CO5: Create lab records of assignments by incorporating problem definitions, design of solutions, results and interpretations.
CO6: Demonstrate the use of ethical practices, self-learning and team spirit.

CO-PO Mapping

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	2	2	2	3	3				3	3	3	3
CO2	3	3	2	3	3	2			3	3	3	2
CO3	3	3	2	3	3	3			3	3	3	2
CO4	3	3	2	3	3	3			3	3	3	3
CO5										3		
CO6								3	3			3

CO-PSO Mapping

CO	PSO1	PSO2	PSO3	PSO4
CO1	2	2	3	
CO2		2	3	
CO3	3	3	3	
CO4	3	3		2
CO5				3
CO6			2	2

*3-HIGH
*2-MEDIUM
*1-LOW

Course Overview

The course "Statistical Machine Learning" introduces students to the fundamental concepts, techniques, and algorithms used in machine learning from a statistical perspective. The course aims to provide students with a solid foundation in both statistical theory and practical applications of machine learning algorithms.

The Statistical Machine Learning Lab is a practical companion course to the Statistical Machine Learning theory course. It provides students with hands-on experience in implementing, experimenting, and analyzing various machine learning algorithms and techniques. The lab sessions are designed to reinforce the concepts learned in the theory course and develop students' skills in applying statistical machine learning methods to real-world problems. Through a combination of programming assignments, data analysis tasks, and mini-projects, students will gain proficiency in using machine learning libraries and tools and gain practical insights into the challenges and nuances of applying machine learning in practice.

List of Experiments mapped with COs

S. No.	Program Name	Course Outcomes
1.	Python program to demonstrate the different operators in python.	CO1
2.	Python program to demonstrate the matrices addition, subtraction, and multiplication.	CO1
3.	Find the mean, median, mode, variance and standard deviation of a list.	CO3
4.	Implement the linear regression algorithm.	CO1, CO2
5.	Implement the logistic regression algorithm.	CO1, CO4
6.	Implement the lasso regression algorithm.	CO1, CO3
7.	Implement the ridge regression algorithm.	CO1, CO2
8.	Implement the K-nearest neighbour algorithm.	CO1, CO2
9.	Implement the decision tree algorithm.	CO1
10.	Implement the random forest algorithm.	CO1, CO4
11.	Implement the naïve bayesian classification algorithm.	CO1, CO3
12.	Implement the support vector machine algorithm.	CO1
13.	Implement the principal component analysis algorithm.	CO1
14.	Implement the singular value decomposition algorithm.	CO1
15.	Implement the k – means clustering algorithm.	CO1, CO2

DOs and DON'Ts

DOs

1. Login-on with your username and password.
2. Log off the Computer every time when you leave the Lab.
3. Arrange your chair properly when you are leaving the lab.
4. Put your bags in the designated area.
5. Ask permission to print.

DON'Ts

1. Do not share your username and password.
2. Do not remove or disconnect cables or hardware parts.
3. Do not personalize the computer setting.
4. Do not run programs that continue to execute after you log off.
5. Do not download or install any programs, games or music on computer in Lab.
6. Personal Internet use chat room for Instant Messaging (IM) and Sites is strictly prohibited.
7. No Internet gaming activities allowed.
8. Tea, Coffee, Water & Eatables are not allowed in the Computer Lab.

General Safety Precautions

Precautions (In case of Injury or Electric Shock)

1. To break the victim with live electric source, use an insulator such as fire wood or plastic to break the contact. Do not touch the victim with bare hands to avoid the risk of electrifying yourself.

2. Unplug the risk of faulty equipment. If main circuit breaker is accessible, turn the circuit off.

3. If the victim is unconscious, start resuscitation immediately, use your hands to press the chest
in and out to continue breathing function. Use mouth-to-mouth resuscitation if necessary.

4. Immediately call medical emergency and security. Remember! Time is critical; be best.

Precautions (In case of Fire)

1. Turn the equipment off. If power switch is not immediately accessible, take plug off.

2. If fire continues, try to curb the fire, if possible, by using the fire extinguisher or by covering it with a heavy cloth if possible, isolate the burning equipment from the other surrounding equipment.

3. Sound the fire alarm by activating the nearest alarm switch located in the hallway.

4. Call security and emergency department immediately:

Emergency : 201 (Reception)
Security: 231 (Gate No.1)

Guidelines to students for report preparation

All students are required to maintain a record of the experiments conducted by them. Guidelines for its preparation are as follows: -

1) All files must contain a title page followed by an index page. ***The files will not be signed by the faculty without an entry in the index page.***
2) Student's Name, roll number and date of conduction of experiment must be written on all pages.
3) For each experiment, the record must contain the following
   (i) Aim/Objective of the experiment
   (ii) Pre-experiment work (as given by the faculty)
   (iii) Lab assignment questions and their solutions
   (iv) Test Cases (if applicable to the course)
   (v) Results/ output

Note:

1. Students must bring their lab record along with them whenever they come for the lab.
2. Students must ensure that their lab record is regularly evaluated.

Lab Assessment Criteria

An estimated 10 lab classes are conducted in a semester for each lab course. These lab classes are assessed continuously. Each lab experiment is evaluated based on 5 assessment criteria as shown in following table. Assessed performance in each experiment is used to compute CO attainment as well as internal marks in the lab course.

Grading Criteria	Exemplary (4)	Competent (3)	Needs Improvement (2)	Poor (1)
AC1: Pre-Lab written work (this may be assessed through viva)	Complete procedure with underlined concept is properly written	Underlined concept is written but procedure is incomplete	Not able to write concept and procedure	Underlined concept is not clearly understood
AC2: Program Writing/ Modeling	Unable to understand the reason for errors/ bugs even after they are explicitly pointed out	Assigned problem is properly analyzed, correct solution designed, appropriate language constructs/ tools are applied	Assigned problem is properly analyzed & correct solution designed	Assigned problem is properly analyzed
AC3: Identification & Removal of errors/ bugs	Able to identify errors/ bugs and remove them	Able to identify errors/ bugs and remove them with little bit of guidance	Is dependent totally on someone for identification of errors/ bugs and their removal	Unable to understand the reason for errors/ bugs even after they are explicitly pointed out
AC4: Execution & Demonstration	All variants of input /output are tested, Solution is well demonstrated and implemented concept is clearly explained	All variants of input /output are not tested, However, solution is well demonstrated and implemented concept is clearly explained	Only few variants of input /output are tested, Solution is well demonstrated but implemented concept is not clearly explained	Solution is not well demonstrated and implemented concept is not clearly explained
AC5: Lab Record Assessment	All assigned problems are well recorded with objective, design constructs and solution along with Performance analysis using all variants of input and output	More than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done with all variants of input and output	Less than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done with all variants of input and output	

LAB EXPERIMENTS

PROGRAM NO. 1

AIM: - Python program to demonstrate the different operators in python.

SOURCE CODE: -

```
# Arithmetic operators
x = 10
y = 5
print("Arithmetic Operators:")
print("x + y =", x + y)  # Addition
print("x - y =", x - y)  # Subtraction
print("x * y =", x * y)  # Multiplication
print("x / y =", x / y)  # Division
print("x % y =", x % y)  # Modulo
print("x ** y =", x ** y)  # Exponentiation
print("x // y =", x // y)  # Floor division

# Comparison operators
a = 7
b = 12
print("\nComparison Operators:")
print("a == b :", a == b)  # Equal to
print("a != b :", a != b)  # Not equal to
print("a > b  :", a > b)   # Greater than
print("a < b  :", a < b)   # Less than
print("a >= b :", a >= b)  # Greater than or equal to
print("a <= b :", a <= b)  # Less than or equal to

# Logical operators
p = True
q = False

print("\nLogical Operators:")
print("p and q:", p and q)  # Logical AND
print("p or q :", p or q)   # Logical OR
print("not p  :", not p)    # Logical NOT

# Assignment operators
```

```
x = 10
y = 5

print("\nAssignment Operators:")
x += y  # Equivalent to x = x + y
print("x =", x)
x -= y  # Equivalent to x = x - y
print("x =", x)
x *= y  # Equivalent to x = x * y
print("x =", x)
x /= y  # Equivalent to x = x / y
print("x =", x)
x %= y  # Equivalent to x = x % y
print("x =", x)
x **= y  # Equivalent to x = x ** y
print("x =", x)
x //= y  # Equivalent to x = x // y
print("x =", x)

# Membership operators
fruits = ['apple', 'banana', 'orange']

print("\nMembership Operators:")
print("'apple' in fruits   :", 'apple' in fruits)    # Check if 'apple' is in the list
print("'grape' not in fruits:", 'grape' not in fruits)  # Check if 'grape' is not in the
list

# Identity operators
x = 10
y = 10
z = 5
print("\nIdentity Operators:")
print("x is y:", x is y)  # Check if x and y refer to the same object
print("x is z:", x is z)  # Check if x and z refer to the same object
print("x is not z:", x is not z)  # Check if x and z refer to different objects
```

OUTPUT:

Arithmetic Operators:
x + y = 15
x - y = 5
x * y = 50
x / y = 2.0
x % y = 0
x ** y = 100000
x // y = 2

Comparison Operators:
a == b : False
a != b : True
a > b : False
a < b : True
a >= b : False
a <= b : True

Logical Operators:
p and q: False
p or q : True
not p : False

Assignment Operators:
x = 15
x = 10
x = 50
x = 10.0
x = 0.0
x = 0.0
x = 0.0

Membership Operators:
'apple' in fruits   : True
'grape' not in fruits: True

Identity Operators:
x is y: True
x is z: False

x is not z: True

PROGRAM NO. 2

AIM: - Python program to demonstrate the matrices addition, subtraction, multiplication.

SOURCE CODE: -

```
import numpy as np

# Matrix addition
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])

C = A + B
print("Matrix Addition:")
print(C)

# Matrix subtraction
D = A - B
print("\nMatrix Subtraction:")
print(D)

# Matrix multiplication
E = A.dot(B)
print("\nMatrix Multiplication:")
print(E)
```

OUTPUT:

Matrix Addition:

```
[[ 6  8]
 [10 12]]
```

Matrix Subtraction:

```
[[-4 -4]
 [-4 -4]]
```

Matrix Multiplication:

```
[[19 22]
 [43 50]]
```

PROGRAM NO. 3

AIM: - Find the mean, median, mode, variance and standard deviation of a list.

SOURCE CODE: -

```
import statistics

data = [1, 2, 3, 4, 5, 5, 6, 7, 8, 9]

# Mean
mean = statistics.mean(data)
print("Mean:", mean)

# Median
median = statistics.median(data)
print("Median:", median)

# Mode
mode = statistics.mode(data)
print("Mode:", mode)

# Variance
variance = statistics.variance(data)
print("Variance:", variance)

# Standard Deviation
std_deviation = statistics.stdev(data)
print("Standard Deviation:", std_deviation)
```

OUTPUT:

Mean: 5.0

Median: 5.0

Mode: 5

Variance: 7.5

Standard Deviation: 2.7386127875258306

PROGRAM NO. 4

AIM: - Implement the linear regression algorithm.

SOURCE CODE: -

```
import numpy as np
from sklearn.linear_model import LinearRegression

# Sample data
X = np.array([[1], [2], [3], [4], [5]])
y = np.array([2, 4, 5, 7, 8])

# Create and fit the model
model = LinearRegression()
model.fit(X, y)

# Predict the output
X_test = np.array([[6], [7], [8]])
y_pred = model.predict(X_test)

# Print the coefficients and intercept
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)

# Print the predictions
print("Predictions:", y_pred)
```

OUTPUT:

Coefficients: [1.2]

Intercept: 0.6

Predictions: [8.4 9.6 10.8]

PROGRAM NO. 5

AIM: - Implement the logistic regression algorithm.

SOURCE CODE: -

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score


# Sample data
X = np.array([[1, 2], [2, 3], [3, 1], [4, 3], [5, 3]])
y = np.array([0, 0, 0, 1, 1])


# Split the 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=42)


# Create and fit the model
model = LogisticRegression()
model.fit(X_train, y_train)
# Predict the output
y_pred = model.predict(X_test)
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

OUTPUT:
Accuracy: 0.5

PROGRAM NO. 6

AIM: - Implement the lasso regression algorithm.

SOURCE CODE: -

```
from sklearn.linear_model import Lasso
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
# Generate synthetic data for demonstration
X, y = make_regression(n_samples=100, n_features=10, random_state=42)
# Split the 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=42)
# Create a Lasso regression model
lasso = Lasso(alpha=0.1)  # Adjust the alpha parameter to control the
regularization strength
# Train the model
lasso.fit(X_train, y_train)
# Make predictions on the test set
y_pred = lasso.predict(X_test)
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

OUTPUT:

Mean Squared Error: 550.3947283733649

PROGRAM NO. 7

AIM: - Implement the ridge regression algorithm.

SOURCE CODE: -

```
import numpy as np
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Sample data
X = np.array([[1, 2], [2, 3], [3, 1], [4, 3], [5, 3]])
y = np.array([5, 6, 8, 9, 11])

# Split the 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=42)

# Create and fit the model
model = Ridge(alpha=0.5)  # Set regularization parameter alpha
model.fit(X_train, y_train)

# Predict the output
y_pred = model.predict(X_test)

# Calculate root mean squared error
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error:", rmse)
```

OUTPUT:

Root Mean Squared Error: 0.608276253029821

PROGRAM NO. 8

AIM: - Implement the K-nearest neighbor algorithm.

SOURCE CODE: -

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Sample data
X = np.array([[3, 5], [4, 7], [6, 3], [7, 4], [8, 2], [9, 6]])
y = np.array([0, 0, 1, 1, 1, 0])

# Split the 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=42)

# Create and fit the model
model = KNeighborsClassifier(n_neighbors=3)  # Set the number of neighbors
model.fit(X_train, y_train)

# Predict the output
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

OUTPUT:

Accuracy: 0.5

PROGRAM NO. 9

AIM: - Implement the decision tree algorithm.

SOURCE CODE: -

```
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Sample data
X = np.array([[1, 1], [1, 0], [0, 1], [0, 0]])
y = np.array([1, 0, 0, 1])

# Split the 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=42)

# Create and fit the model
model = DecisionTreeClassifier()
model.fit(X_train, y_train)

# Predict the output
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

OUTPUT:

Accuracy: 0.6

PROGRAM NO. 10

AIM: - Implement the random forest algorithm.

SOURCE CODE: -

```
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Sample data
X = np.array([[1, 1], [1, 0], [0, 1], [0, 0]])
y = np.array([1, 0, 0, 1])

# Split the 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=42)

# Create and fit the model
model = RandomForestClassifier(n_estimators=100)  # Set the number of decision trees
model.fit(X_train, y_train)

# Predict the output
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

OUTPUT:

Accuracy: 0.767

PROGRAM NO. 11

AIM: - Implement the naïve Bayesian classification algorithm.

SOURCE CODE: -

```
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Sample data
X = np.array([[1, 2], [2, 3], [3, 1], [4, 3], [5, 3], [6, 2]])
y = np.array([0, 0, 1, 1, 1, 0])

# Split the 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=42)

# Create and fit the Naive Bayes classifier
model = GaussianNB()
model.fit(X_train, y_train)

# Predict the output
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

OUTPUT:

Accuracy: 0.82

PROGRAM NO. 12

AIM: - Implement the support vector machine algorithm.

SOURCE CODE: -

```
from sklearn.svm import SVC
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Generate synthetic data for demonstration
X, y = make_classification(n_samples=100, n_features=10, random_state=42)

# Split the 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=42)

# Create an SVM classifier
svm = SVC(kernel='linear')  # You can choose different kernel functions like 'linear', 'rbf',
'poly', etc.

# Train the classifier
svm.fit(X_train, y_train)

# Make predictions on the test set
y_pred = svm.predict(X_test)

# Evaluate the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

OUTPUT:

Accuracy: 0.9

PROGRAM NO. 13

AIM: - Implement the principal component analysis algorithm.

SOURCE CODE: -

```
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris

# Load the Iris dataset for demonstration
data = load_iris()
X = data.data
y = data.target

# Create a PCA object and specify the number of components
pca = PCA(n_components=2)  # You can choose the desired number of
components

# Apply PCA to the data
X_pca = pca.fit_transform(X)

# Access the principal components
principal_components = pca.components_

# Access the explained variance ratio
explained_variance_ratio = pca.explained_variance_ratio_
# Print the results
print("Principal Components:")
print(principal_components)
print("Explained Variance Ratio:")
print(explained_variance_ratio)
```

OUTPUT:

Principal Components:

[[0.36158968 -0.08226889 0.85657211 0.35884393]

 [-0.65653988 -0.72971237 0.1757674 -0.07470647]]

Explained Variance Ratio:

[0.92461621 0.05301557]

PROGRAM NO. 14

AIM: - Implement the singular value decomposition algorithm.

SOURCE CODE: -

```
import numpy as np

# Create a sample matrix
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# Perform Singular Value Decomposition
U, S, VT = np.linalg.svd(A)

# Print the singular values
print("Singular values:")
print(S)

# Print the left singular vectors (U)
print("Left singular vectors (U):")
print(U)

# Print the right singular vectors (VT)
print("Right singular vectors (VT):")
print(VT)
```

OUTPUT:

Singular values:

[1.68481034e+01 1.06836951e+00 1.47280877e-16]

Left singular vectors (U):

[[-0.21483724 0.88723069 0.40824829]

 [-0.52058739 0.24964395 -0.81649658]

 [-0.82633753 -0.3879428   0.40824829]]

Right singular vectors (VT):

[[-0.47967185 -0.57236779 -0.66506372]

 [-0.77669099 -0.07568647 0.62431804]

 [0.40824829 -0.81649658 0.40824829]]

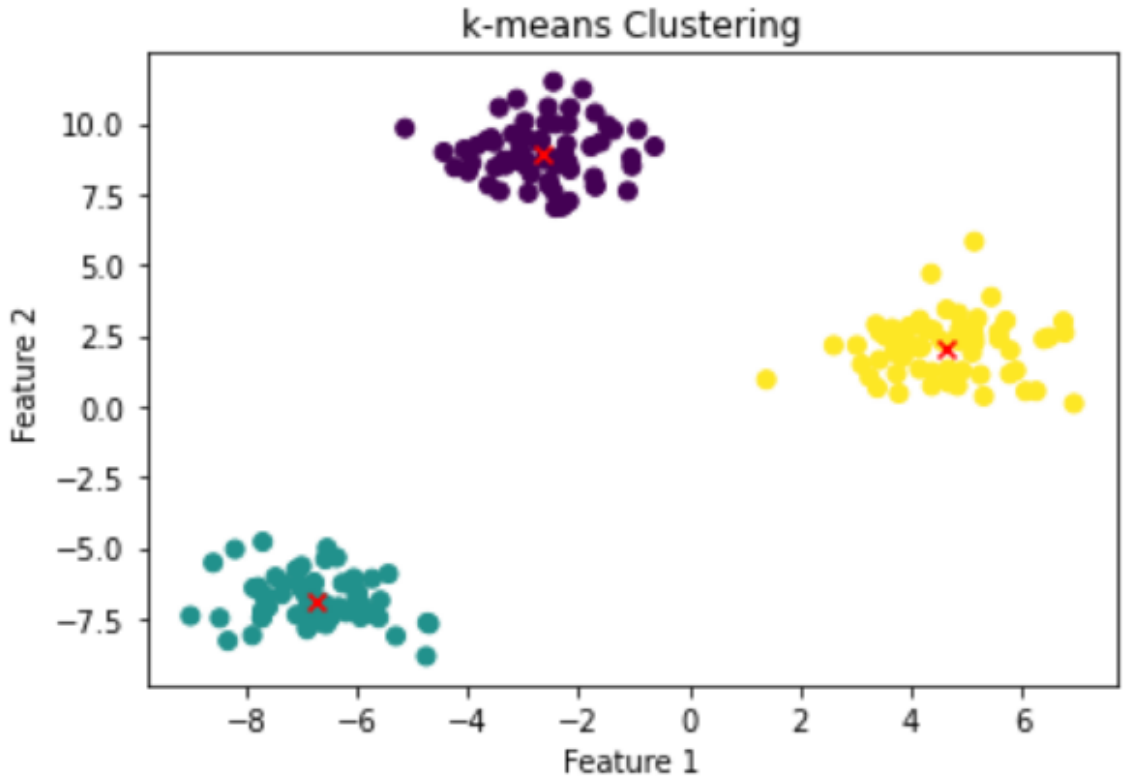
PROGRAM NO. 15

AIM: - Implement the k – means clustering algorithm.

SOURCE CODE: -

```
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt

# Generate synthetic data for demonstration
X, y = make_blobs(n_samples=200, centers=3, random_state=42)

# Create a KMeans clustering object and specify the number of clusters
kmeans = KMeans(n_clusters=3)  # You can choose the desired number of clusters

# Perform clustering on the data
kmeans.fit(X)

# Get the cluster labels and cluster centers
labels = kmeans.labels_
centers = kmeans.cluster_centers_

# Visualize the clusters
plt.scatter(X[:, 0], X[:, 1], c=labels)
plt.scatter(centers[:, 0], centers[:, 1], marker='x', color='red')
plt.title("k-means Clustering")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```

OUTPUT:


k-means Clustering
Feature 2
Feature 1
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−8
−6
−4
−2
0
2
4
6

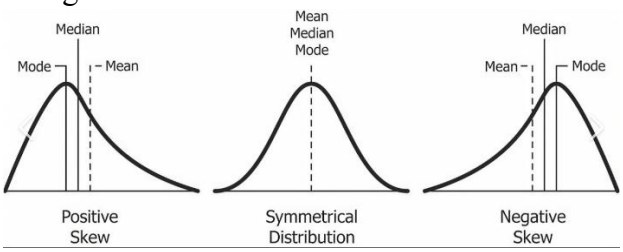
Viva – Voice Questions

1. What are Skewness in statistics and its types?

Skewness is a measure of the symmetry of a distribution. A distribution is symmetrical if it is shaped like a bell curve, with most of the data points concentrated around the mean. A distribution is skewed if it is not symmetrical, with more data points concentrated on one side of the mean than the other.

There are two types of skewness: positive skewness and negative skewness.

- **Positive skewness:** Positive skewness occurs when the distribution has a long tail on the right side, with the majority of the data points concentrated on the left side of the mean. Positive skewness indicates that there are a few extreme values on the right side of the distribution that is pulling the mean to the right.
- **Negative skewness:** Negative skewness occurs when the distribution has a long tail on the left side, with the majority of the data points concentrated on the right side of the mean. Negative skewness indicates that there are a few extreme values on the left side of the distribution that is pulling the mean to the left.



2. What are the measures of central tendency?

In statistics, measures of central tendency are values that represent the center of a dataset. There are three main measures of central tendency: mean, median, and mode.

- The mean is the arithmetic average of a dataset and is calculated by adding all the values in the dataset and dividing by the number of values. The mean is sensitive to outliers, or values that are significantly higher or lower than the majority of the other values in the dataset.
- The median is the middle value of a dataset when the values are arranged in order from smallest to largest. To find the median, you must first arrange the values in order and then locate the middle value. If there is an odd number of values, the median is the middle value. If there is an even number of values, the median is the mean of the two middle values. The median is not sensitive to outliers.
- The mode is the value that occurs most frequently in a dataset. A dataset may have multiple modes or no modes at all. The mode is not sensitive to outliers.

3. Can you explain the concept of correlation and covariance?

Correlation is a statistical measure that describes the strength and direction of a linear relationship between two variables. A positive correlation indicates that the two variables increase or decrease together, while a negative correlation indicates that the two variables

move in opposite directions. Covariance is a measure of the joint variability of two random variables. It is used to measure how two variables are related.

4. What is the significance of the p-value?

The p-value is used to determine the statistical significance of a result. In hypothesis testing, the p-value is used to assess the probability of obtaining a result that is at least as extreme as the one observed, given that the null hypothesis is true. If the p-value is less than the predetermined level of significance (usually denoted as alpha, α), then the result is considered statistically significant and the null hypothesis is rejected.

The significance of the p-value is that it allows researchers to make decisions about the data based on a predetermined level of confidence. By setting a level of significance before conducting the statistical test, researchers can determine whether the results are likely to have occurred by chance or if there is a real effect present in the data.

5. What is the difference between a parametric and a non-parametric test?

A parametric test is a statistical test that assumes that the data follows a specific probability distribution, such as a normal distribution. A non-parametric test does not make any assumptions about the underlying probability distribution of the data.

6. What are the 5 assumptions for linear regression?

Here are the 5 assumptions of linear regression:

1. Linearity: There is a linear relationship between the independent variables and the dependent variable.
2. Independence of errors: The errors (residuals) are independent of each other.
3. Homoscedasticity: The variance of the errors is constant across all predicted values.
4. Normality: The errors follow a normal distribution.
5. Independence of predictors: The independent variables are not correlated with each other.

7. What is the difference between linear and nonlinear regression?

- **Linear regression** is the method in which is used to find the relationship between a dependent and one or more independent variables. The model finds the best-fit line, which is a linear function (y = mx +c) that helps in fitting the model in such a way that the error is minimum considering all the data points. So, the decision boundary of a linear regression function is linear.
- **A non-Linear regression** is used to model the relationship between a dependent and one or more independent variables by a non-linear equation. The non-linear regression models are more flexible and are able to find the more complex relationship between variables.

8. How will you identify underfitting in a model?

Underfitting occurs when a statistical model or machine learning algorithm is not able to capture the underlying trend of the data. This can happen for a variety of reasons, but one common cause is that the model is too simple and is not able to capture the complexity of the data

Here is how to identify underfitting in a model:

- The **training error** of an underfitting error will be high, i.e., the model will not be able to learn from the training data and will perform poorly on the training data.
- The **validation error** of an underfitting model will also be high as it will perform poorly on the new data as well.

9. How will you identify overfitting in a model?

Overfitting in a model occurs when the model learns the whole training data instead of taking signals/hints from the data and the model performs extremely well on training data and performs poorly on the testing data. The testing error of the model is high compared to the training error. The bias of an overfitting model is low whereas the variance is high.

10. What is Multicollinearity?

Multicollinearity occurs when two or more predictor variables in a multiple regression model are highly correlated. This can lead to unstable and inconsistent coefficients, and make it difficult to interpret the results of the model. Multicollinearity occurs when there is a high degree of correlation between two or more predictor variables. This can make it difficult to determine the unique contribution of each predictor variable to the response variable, as the estimates of their coefficients may be influenced by the other correlated variables.

11. What is the difference between K-means and KNN?

K-means and KNN (K-Nearest Neighbors) are two different machine learning algorithms.

- **K-means** is a clustering algorithm that is used to divide a group of data points into K clusters, where each data point belongs to the cluster with the nearest mean. It is an iterative algorithm that assigns data points to a cluster and then updates the cluster centroid (mean) based on the data points assigned to it.
- **KNN** is a classification algorithm that is used to classify data points based on their similarity to other data points. It works by finding the K data points in the training set that are most similar to the data point being classified, and then it assigns the data point to the class that is most common among those K data points.

So, in summary, K-means is used for clustering, and KNN is used for classification.

12. How do you decide whether a model is suffering from high bias or high variance?

There are several ways to determine whether a model is suffering from high bias or high variance. Some common methods are:

- Split the data into a training set and a test set, and check the performance of the model on both sets. If the model performs well on the training set but poorly on the test set, it is likely to suffer from high variance (overfitting). If the model performs poorly on both sets, it is likely suffering from high bias (underfitting).
- Use cross-validation to estimate the performance of the model. If the model has high variance, the performance will vary significantly depending on the data used for training and testing. If the model has high bias, the performance will be consistently low across different splits of the data.
- Plot the learning curve, which shows the performance of the model on the training set and the test set as a function of the number of training examples. A model with high

bias will have a high training error and a high-test error, while a model with high variance will have a low training error and a high-test error.

13. What are some techniques for balancing bias and variance in a model?

There are several techniques that can be used to balance the bias and variance in a model, including:

Increasing the model complexity by adding more parameters or features: This can help the model capture more complex patterns in the data and reduce bias, but it can also increase variance if the model becomes too complex.

- Reducing the model complexity by removing parameters or features: This can help the model avoid overfitting and reduce variance, but it can also increase bias if the model becomes too simple.
- Using regularization techniques: These techniques constrain the model complexity by penalizing large weights, which can help the model avoid overfitting and reduce variance. Some examples of regularization techniques are L1 regularization, L2 regularization, and elastic net regularization.
- Splitting the data into a training set and a test set: This allows us to evaluate the model's generalization ability and tune the model complexity to achieve a good balance between bias and variance.
- Using cross-validation: This is a technique for evaluating the model's performance on different splits of the data and averaging the results to get a more accurate estimate

of the model's generalization ability.

14. How do you choose the appropriate evaluation metric for a classification problem, and how do you interpret the results of the evaluation?

There are many evaluation metrics that you can use for a classification problem, and the appropriate metric depends on the specific characteristics of the problem and the goals of the evaluation. Some common evaluation metrics for classification include:

- **Accuracy:** This is the most common evaluation metric for classification. It measures the percentage of correct predictions made by the model.
- **Precision:** This metric measures the proportion of true positive predictions among all positive predictions made by the model.
- **Recall:** This metric measures the proportion of true positive predictions among all actual positive cases in the test set.
- **F1 Score:** This is the harmonic mean of precision and recall. It is a good metric to use when you want to balance precision and recall.
- **AUC-ROC:** This metric measures the ability of the model to distinguish between positive and negative classes. It is commonly used for imbalanced classification problems.

To interpret the results of the evaluation, you should consider the specific characteristics of the problem and the goals of the evaluation. For example, if you are trying to identify fraudulent transactions, you may be more interested in maximizing precision, because you want to minimize the number of false alarms. On the other hand, if you are trying to diagnose a disease, you may be more interested in maximizing recall, because you want to minimize the number of missed diagnoses.

15. When not to use PCA for dimensionality reduction?

There are several situations when you may not want to use Principal Component Analysis (PCA) for dimensionality reduction:

- When the data is not linearly separable: PCA is a linear technique, so it may not be effective at reducing the dimensionality of data that is not linearly separable.
- When the data has categorical features: PCA is designed to work with continuous numerical data and may not be effective at reducing the dimensionality of data with categorical features.
- When the data has a large number of missing values: PCA is sensitive to missing values and may not work well with data sets that have a large number of missing values.
- When the data is highly imbalanced: PCA is sensitive to class imbalances and may not produce good results on highly imbalanced data sets.
- When the goal is to preserve the relationships between the original features: PCA is a technique that looks for patterns in the data and creates new features that are combinations of the original features. As a result, it may not be the best choice if the goal is to preserve the relationships between the original features.

16. What is Gradient descent?

Gradient descent is an optimization algorithm used in machine learning to find the values of parameters (coefficients and bias) of a model that minimize the cost function. It is a first-order iterative optimization algorithm that follows the negative gradient of the cost function to converge to the global minimum.

In gradient descent, the model's parameters are initialized with random values, and the algorithm iteratively updates the parameters in the opposite direction of the gradient of the cost function with respect to the parameters. The size of the update is determined by the learning rate, which is a hyperparameter that controls how fast the algorithm converges to the global minimum.

17. What are some common methods for hyperparameter tuning?

There are several common methods for hyperparameter tuning:

- Grid Search: This involves specifying a set of values for each hyperparameter, and the model is trained and evaluated using a combination of all possible hyperparameter values. This can be computationally expensive, as the number of combinations grows exponentially with the number of hyperparameters.
- Random Search: This involves sampling random combinations of hyperparameters and training and evaluating the model for each combination. This is less computationally intensive than grid search, but may be less effective at finding the optimal set of hyperparameters.