Shikshana Prasaraka Mandali's

Sir Parashurambhau College (Autonomous), Pune – 411030

Department of Computer Science

Master of Computer Science
(M.Sc. (Computer Science))
Semester III

MSCOSDSC305 - Practical course based on Machine Learning
(MSCOSDSC302)

PRN/Roll No.:	Name of Student:
Division:	Academic Year: 20 :20

Lab book compiled by: Dr. Madhuri Deshpande
Member, Board of Studies, Computer Science, S.P. College, Pune

Shikshana Prasaraka Mandali's

Sir Parashurambhau College (Empowered Autonomous), Pune – 411030

Department of Computer Science

CERTIFICATE

This is to certify that Mr./Ms _____ having exam Seat Number _____ has successfully completed the assignments for the practical course based on the major subject of MSCOSDSC305 - Practical course based on Machine Learning, during the academic year_____ and has scored _____ marks out of 20.

Teacher Incharge

Head
Dept. of Computer Science

Internal Examiner

External Examiner

Assignment Completion Sheet

Sr. No.	Description	Marks out of 5	Sign
1.	Implementation of Python Basic Libraries such as Statistics, Math, Numpy and Scipy required for ML		
2.	Implementation of Python Libraries for ML application such as Pandas and Matplotlib.		
3.	Creation and loading different types of datasets in Python using the required libraries. Write a python program to compute Mean, Median, Mode, Variance, Standard Deviation using Datasets Demonstrate various data pre-processing techniques for a given dataset.		
4.	Neural network implementation and hyper parameter tuning		
5.	Implement the Find-S and Candidate Elimination (aka List-Then-Eliminate) algorithm on the data sets.		
6.	Implement Gaussian Naïve Bayes Classifier – dataset /kaggle/input/adult-dataset/adult.csv that has 15 attributes, segregate the dataset into categorical and numerical variables		
7.	Implement Multinomial Naive Bayes Classifier- Design and implement a Multinomial Naive Bayes Classifier to classify documents into pre-defined types based on likelihood of a word occurring by using Bayes theorem. The data set shall be provided as a CSV. The dataset will be of text data categorized into four labels: **Technology**, **Sports**, **Politics** and **Entertainment**. Each entry contains a short sentence or statement related to a specific topic with the label indicating the category it belongs to.		
8.	Implement Naïve Bayes Optimal Classifier		
9.	Implement Random Forest algorithm on MNIST data		
10.	Implement Random Forest algorithm on Mental health data		
11.	Implement Random Forest algorithm on customers' default payments data set. Given a set of features, predict the probability of a customer defaulting on a loan. The target variable is "default payment" (Yes=1; No=1)		
12.	Implement k-nearest neighbours classification.		
	Marks out of 60		
	Marks out of 20		

Sign-Teacher Incharge

Introduction

About the Lab Book

This lab book is intended to be used by M. Sc. (Computer Science) students of S.P. College, Pune, for the Practical course based on Machine Learning (MSCOSDSC302).

Machine Learning (ML) practical's are crucial because they transform theoretical understanding into applied intelligence by engaging students in real-world data analysis, model creation, and algorithmic reasoning. Practical sessions help bridge the gap between conceptual learning and the technical proficiency needed in professional and research environments.

ML practical exercises allow students to implement supervised and unsupervised learning algorithms, test performance metrics, and visualize model outcomes. This hands-on approach consolidates classroom concepts such as bias–variance tradeoff, overfitting, and cross-validation.They cultivate critical computational and analytical skills. By engaging with datasets, students gain experience in data preprocessing, model tuning, and interpretation—preparing them to tackle complex, unstructured problems systematically.

Experimentation with various learning paradigms encourages creativity and innovation. Students learn to design solutions using ensemble methods, dimensionality reduction techniques, and instance-based models, fostering higher-order thinking skills aligned with Bloom's levels of analysis, evaluation, and creation.

A practical exposure to ML tools and libraries in Python using experiential learning improves career readiness and expands opportunities in AI-driven domains. In essence, ML practicals nurture applied intelligence by combining theory, computation, creativity, and ethical awareness—enabling students to progress understanding to creation.

Objectives:

1. Equip students with comprehensive knowledge of machine learning principles, including paradigms, key algorithms, and real-world applications.
2. Provide foundational and advanced tools to formulate and solve classification, regression, and clustering problems using modern ML approaches.

Course Outcomes

Outcome Code	Course Outcome	Bloom's Level	Attainment Number
CO1	Implement basic supervised and unsupervised ML algorithms such as Decision Trees, Naïve Bayes, and k-Means.	Apply (Level 3)	2
CO2	Analyze model performance using k-fold cross-validation and bootstrap methods.	Analyze (Level 4)	3
CO3	Evaluate ML models in terms of bias–variance trade-offs and generalization.	Evaluate (Level 5)	3
CO4	Design ensemble and dimensionality reduction solutions such as Bagging, Boosting, and PCA.	Create (Level 6)	3
CO5	Interpret and communicate results through visualization and analytical reporting.	Understand / Apply (Level 2)	2

Instructions to the students:

Please read the following instructions carefully and follow them.

- Students are expected to carry this lab/workbook during every practical.
- Students should prepare oneself beforehand for the assignment by reading the relevant material.
- Teacher/Instructor will specify which problems to solve in the lab during the allotted slot and student should complete them and get verified by the instructor. However, student should spend additional hours in Lab and at home to cover as many problems as possible given in this work book.
- Students will be assessed for each exercise on a scale from 0 to 5

Scale	Remark / Meaning
0	Not done
1	Incomplete
2	Late Complete
3	Needs improvement
4	Complete
5	Well Done.

Assignment 1: Implementation of Python Basic Libraries

Implement basic functions of Statistics and Maths using Numpy and Scipy required for ML.

a) Usage of methods such as floor(), ceil(), sqrt(), isqrt(), gcd() etc.

1. math.floor(x):
   Returns the floor of x, which is the largest integer less than or equal to x.

```
import math
print(math.floor(3.7))  # Output: 3
print(math.floor(-2.3)) # Output: -3
```

2. math.ceil(x):
   Returns the ceiling of x, which is the smallest integer greater than or equal to x.

```
import math
print(math.ceil(3.2))   # Output: 4
print(math.ceil(-2.7))  # Output: -2
```

3. math.sqrt(x):
   Returns the square root of x as a float. x must be non-negative.

```
import math
print(math.sqrt(25))    # Output: 5.0
print(math.sqrt(2))     # Output: 1.4142135623730951
```

4. math.isqrt(n):
   Returns the integer square root of n, which is the largest integer a such that a*a <= n. This function is available from Python 3.8 onwards.

```
import math
print(math.isqrt(25))   # Output: 5
print(math.isqrt(26))   # Output: 5
print(math.isqrt(0))    # Output: 0
```

5. math.gcd(a, b):
   Returns the greatest common divisor (GCD) of the two integers a and b.

```
import math
print(math.gcd(48, 18)) # Output: 6
print(math.gcd(17, 5))  # Output: 1
```

Check other functions from the Math library

b) Usage of attributes of array such as ndim, shape, size, methods such as sum(), mean(), sort(), sin() etc.

c) Usage of methods such as det(), eig() etc.
In Python, the det() and eig() functions, used for calculating the determinant and eigenvalues/eigenvectors of a matrix, are found within the numpy.linalg module.
numpy.linalg.det()
This function computes the determinant of a square matrix.

```
import numpy as np
# Create a square matrix
matrix = np.array([[1, 2],
                   [3, 4]])

# Calculate the determinant
determinant = np.linalg.det(matrix)
print(f"The determinant of the matrix is: {determinant}")
```

numpy.linalg.eig()

This function computes the eigenvalues and right eigenvectors of a square matrix. It returns a tuple containing two arrays: the first array holds the eigenvalues, and the second array holds the corresponding eigenvectors (as columns).

```
import numpy as np

# Create a square matrix
matrix = np.array([[2, 2],
                   [1, 3]])

# Calculate eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(matrix)
print(f"Eigenvalues: {eigenvalues}")
print(f"Eigenvectors:\n{eigenvectors}")
```

d) Consider a list datatype(1D) then reshape it into2D, 3D matrix using numpy

To reshape a 1D Python list into 2D and 3D matrices using NumPy, the list must first be converted into a NumPy array. The reshape() method can then be applied to this array, specifying the desired dimensions.

```
import numpy as np

# Create a 1D Python list
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

# Convert the list to a NumPy array
np_array = np.array(my_list)
print("Original 1D NumPy Array:")
print(np_array)
print("Shape:", np_array.shape)
print("\n")

# Reshape into a 2D matrix (e.g., 3 rows, 4 columns)
# The total number of elements must remain consistent (3 * 4 = 12)
matrix_2d = np_array.reshape(3, 4)
print("Reshaped 2D Matrix:")
print(matrix_2d)
print("Shape:", matrix_2d.shape)
print("\n")

# Reshape into a 3D matrix (e.g., 2 layers, 2 rows, 3 columns)
# The total number of elements must remain consistent (2 * 2 * 3 = 12)
matrix_3d = np_array.reshape(2, 2, 3)
print("Reshaped 3D Matrix:")
print(matrix_3d)
print("Shape:", matrix_3d.shape)
```

Convert to NumPy Array: np.array(my_list) converts the Python list into a NumPy ndarray.
This allows the use of NumPy's array manipulation functions like reshape().

Reshape to 2D: np_array.reshape(3, 4) transforms the 1D array into a 2D array with 3 rows and 4 columns. The product of the new dimensions (3 * 4 = 12) must equal the total number of elements in the original 1D array.

Reshape to 3D: np_array.reshape(2, 2, 3) transforms the 1D array into a 3D array with 2 "layers" (or depth), 2 rows, and 3 columns. Again, the product of the new dimensions (2 * 2 * 3 = 12) must match the total number of elements.

The shape attribute is used to confirm the dimensions of the resulting arrays.

e) Numpy.random.Generator and matrices using numpy

The numpy.random.Generator is the preferred way to generate random numbers in NumPy, offering a more robust and flexible approach compared to older functions like numpy.random.rand() or numpy.random.randint(). It allows for better control over the random number generation process, including the ability to specify a seed for reproducibility and to choose from various distributions.

Using numpy.random.Generator:

Create a Generator instance.

```
import numpy as np
rng = np.random.default_rng(seed=42) # Optional: set a seed for reproducibility
```

Generate random numbers: The Generator object provides methods for generating random numbers from different distributions.

Uniform distribution (floats between 0 and 1):

```
random_floats = rng.random(size=(2, 3)) # Generates a 2x3 matrix of floats
```

Integers within a range.

```
random_integers = rng.integers(low=1, high=11, size=(3, 3))
# Generates a 3x3 matrix of integers between 1 and 10
```

Normal (Gaussian) distribution.

```
normal_values = rng.normal(loc=0, scale=1, size=(2, 2))
# Generates a 2x2 matrix from a standard normal distribution
```

Matrices using NumPy:

NumPy arrays are the fundamental data structure for representing matrices. Creating a matrix.

```
matrix_a = np.array([[1, 2, 3],
                     [4, 5, 6],
                     [7, 8, 9]])
```

Matrix operations: NumPy provides efficient functions for various matrix operations.

Matrix multiplication: Use np.dot() or the @ operator (for Python 3.5+).

```
matrix_b = np.array([[9, 8, 7],
                     [6, 5, 4],
                     [3, 2, 1]])
product = np.dot(matrix_a, matrix_b)
# or
product_at = matrix_a @ matrix_b
```

Transpose.

```
transpose_a = matrix_a.T
```

Element-wise operations: Standard arithmetic operators perform element-wise operations.

```
sum_matrices = matrix_a + matrix_b
```

f) Find the determinant of a matrix using scipy

To find the determinant of a matrix using SciPy, the scipy.linalg.det() function is employed. This function calculates the determinant of a square matrix.

```
import numpy as np
from scipy import linalg
# Define a square matrix
matrix_A = np.array([[3, 1, 4],
                     [1, 5, 9],
                     [2, 6, 5]])
# Calculate the determinant
determinant_A = linalg.det(matrix_A)
print("Matrix A:")
print(matrix_A)
print("\nDeterminant of Matrix A:", determinant_A)
```

g) Find eigen value and eigen vector of a matrix using scipy

Eigenvalues and eigenvectors of a matrix can be found using the eig function from the scipy.linalg module.

Steps:

1. Import necessary libraries: Import numpy to create the matrix and scipy.linalg.eig to calculate eigenvalues and eigenvectors.
2. Define the matrix: Create a square matrix using numpy.array().
3. Calculate eigenvalues and eigenvectors: Call the eig() function, passing the matrix as an argument. This function returns two arrays: the first contains the eigenvalues, and the second contains the eigenvectors. Each column of the eigenvector array corresponds to an eigenvector associated with the eigenvalue at the same index in the eigenvalue array.

Example:

```
import numpy as np
from scipy.linalg import eig

# Define a square matrix
A = np.array([[2, 1],
              [1, 2]])

# Calculate eigenvalues and eigenvectors
eigenvalues, eigenvectors = eig(A)
# Print the results
print("Eigenvalues:", eigenvalues)
print("Eigenvectors:\n", eigenvectors)
```

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well done [ ]

Signature of the instructor and date

Assignment 2: Implementation of Pandas and matplotlib

Implementation of Python Libraries for ML application such as Pandas and Matplotlib.

a) Create a Series using pandas and display
b) Access the index and the values of our Series
c) Compare an array using Numpy with a series using pandas
d) Define Series objects with individual indices
e) Access single value of a series
f) Load datasets in a Data frame variable using pandas
   Usage of different methods in Matplotlib.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well done [ ]

Signature of the instructor and date

Assignment 3: Creation and loading different types of datasets

Data set Creation:

i. Creation using pandas

Creating datasets using Pandas primarily involves constructing a DataFrame, which is a two-dimensional, tabular data structure with labeled axes (rows and columns). There are several common methods to achieve this:

1. From a Dictionary:

A common way to create a DataFrame is from a dictionary where keys represent column names and values are lists or NumPy arrays containing the data for each column.

```
import pandas as pd
data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [25, 30, 22],
        'City': ['New York', 'London', 'Paris']}
df = pd.DataFrame(data)
print(df)
```

2. From a List of Lists:

You can create a DataFrame from a list of lists, where each inner list represents a row of data. You can optionally provide column names.

```
import pandas as pd
data = [['Alice', 25, 'New York'],
        ['Bob', 30, 'London'],
        ['Charlie', 22, 'Paris']]
df = pd.DataFrame(data, columns=['Name', 'Age', 'City'])
print(df)
```

3. From a List of Dictionaries:

This method is useful when each row of your data is represented as a dictionary.

```
import pandas as pd
data = [{'Name': 'Alice', 'Age': 25, 'City': 'New York'},
        {'Name': 'Bob', 'Age': 30, 'City': 'London'},
        {'Name': 'Charlie', 'Age': 22, 'City': 'Paris'}]
df = pd.DataFrame(data)
print(df)
```

4. From External Files (CSV, Excel, etc.):

Pandas offers powerful functions to read data directly from various file formats into a DataFrame.

```
import pandas as pd
# From a CSV file
df_csv = pd.read_csv('data.csv')
print(df_csv.head())

# From an Excel file
df_excel = pd.read_excel('data.xlsx')
print(df_excel.head())
```

5. From a NumPy Array:
You can create a DataFrame from a NumPy array, optionally providing column names and an index.

```
import pandas as pd
import numpy as np
data = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
df_np = pd.DataFrame(data, columns=['ColA', 'ColB', 'ColC'])
print(df_np)
```

ii. Loading CSV dataset files using Pandas

```
import pandas as pd
df = pd.read_csv('/content/drive/MyDrive/your_folder/your_file.csv')
```

iii. Loading datasets using sklearn

```
from sklearn.datasets import load_iris, load_digits, load_diabetes

# Load the Iris dataset
iris = load_iris()
X_iris, y_iris = iris.data, iris.target
print(X_iris)

# Load the Digits dataset
digits = load_digits()
X_digits, y_digits = digits.data, digits.target

# Load the Diabetes dataset
diabetes = load_diabetes()
X_diabetes, y_diabetes = diabetes.data, diabetes.target
```

iv. Loading data sets into Google Colab

```
from google.colab import files
uploaded = files.upload()
```

b) Write a python program to compute Mean, Median, Mode, Variance, Standard Deviation using Datasets

c) Demonstrate various data pre-processing techniques for a given dataset. Write a python program to compute
   i. Reshaping the data
   ii. Filtering the data
   iii. Merging the data
   iv. Handling the missing values in datasets
   v. Feature Normalization: Min-max normalization, Scalar Normalization etc.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well done [ ]

Signature of the instructor and date

Assignment 4: Implementing Neural Networks

An artificial neural network is a biologically inspired computational model that is patterned after the network of neurons present in the human brain. Artificial neural networks can also be thought of as learning algorithms that model the input-output relationship. Applications of artificial neural networks include pattern recognition and forecasting in fields such as medicine, business, pure sciences, data mining, telecommunications, and operations managements.

An artificial neural network transforms input data by applying a nonlinear function to a weighted sum of the inputs. The transformation is known as a neural layer and the function is referred to as a neural unit. The intermediate outputs of one layer, called features, are used as the input into the next layer. The neural network through repeated transformations learns multiple layers of nonlinear features (like edges and shapes), which it then combines in a final layer to create a prediction (of more complex objects).

The neural net learns by varying the weights or parameters of a network so as to minimize the difference between the predictions of the neural network and the desired values. This phase where the artificial neural network learns from the data is called training.
All ANNs are very good classifiers.

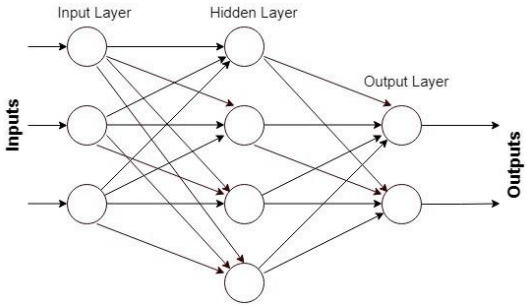
How do neural networks work?
Neural networks are composed of a collection of nodes. An artificial neural network is made of artificial neurons that work together to solve a problem. Artificial neurons are software modules, called nodes, and artificial neural networks are software programs or algorithms that, at their core, use computing systems to solve mathematical calculations. The nodes are spread out across at least three layers.

Simple neural network architecture
A basic neural network has interconnected artificial neurons in three layers:

1. Input Layer
   Information from the outside world enters the artificial neural network from the input layer. Input nodes process the data, analyze or categorize it, and pass it on to the next layer
2. Hidden Layer
   Hidden layers take their input from the input layer or other hidden layers. Artificial neural networks can have a large number of hidden layers. Each hidden layer analyzes the output from the previous layer, processes it further, and passes it on to the next layer.
3. Output Layer
   The output layer gives the final result of all the data processing by the artificial neural network. It can have single or multiple nodes. For instance, if we have a binary (yes/no) classification problem, the output layer will have one output node, which will give the result as 1 or 0. However, if we have a multi-class classification problem, the output layer might consist of more than one output node.

Types of Neural Networks

Feedforward Neural Networks



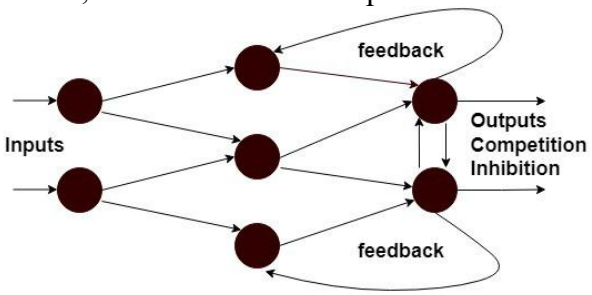
A feedforward neural network, information flow starts from the input layer to the hidden layer(s) and finally to the output layer. These neural networks are simple and have a one-to-one mapping between inputs with outputs.

Feedback Neural Networks

In this type of networks, the signal or the information flows in both directions, i.e., forward and backward. This makes them more powerful and more complex than the feed-forward neural networks.

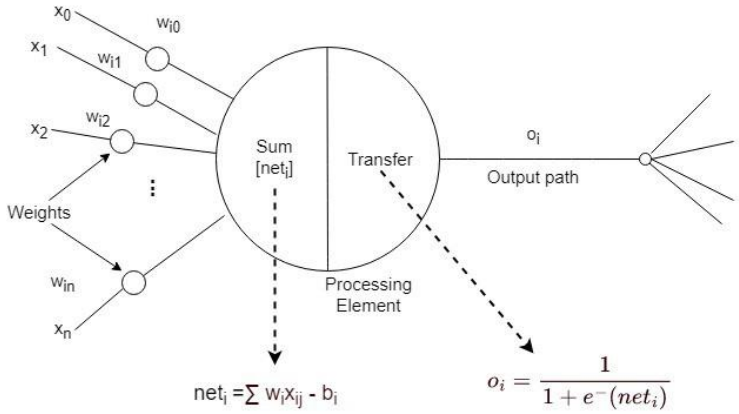
Feedback neural networks are dynamic because the network state keeps on changing until it reaches an equilibrium point. They remain at the equilibrium point till the input remains the same. Once the input changes, this process goes on until they find a new equilibrium.

We also call these networks interactive or recurrent networks due to their dynamic architecture. Moreover, we find feedback loops in these networks.



Key terms:

- **Neuron/Processing Element.** A Neuron is an information processing unit in a neural network. Each neuron processes some input by applying an "Activation Function" and serves the result of the activation function as its output.



- **Perceptron.** A perceptron is a neuron that takes binary inputs and produces a single

binary output.

- **Activation Function.** The function that we pass the input information through in a neuron.

https://www.v7labs.com/blog/neural-networks-activation-functions

- **Epoch.** One complete pass of the entire training dataset through the learning algorithm. hey are directly related to how well a model learns and generalizes to unseen data. The number of epochs is a hyperparameter that defines the number times that the learning algorithm will work through the entire training dataset. Too few epochs can result in an underfit model, whereas too many epochs can lead to overfitting.
- **Epochs vs. Iterations vs. Batches**

    - Batch: A set of N samples from the dataset. The batch size is a hyperparameter that determines the number of samples to work through before updating the internal model parameters.
    - Iteration: One update of the model's parameters. Each iteration is the number of batches needed to complete one epoch.
    - Epoch: One full cycle through the training data, as described above.
- **Bias.** Additional input given to each neuron, bias helps in controlling the value at which activation function will trigger. It is used to offset the result. It helps the models to shift the activation function towards the positive or negative side. If bias is not used, neural network will train over a point passing through origin only, which is not in accordance with real-world scenario.
- **Sigmoid.** A common activation function. S-shaped curve that ranges between 0 and 1. Neurons are sometimes referred to as "sigmoid neurons," meaning they are neurons that use the sigmoid activation function.
- **Tanh.** A common activation function. S-shaped curve that ranges between -1 and 1. Neurons are sometimes referred to as "sigmoid neurons," meaning they are neurons that use the sigmoid activation function. Tanh is used more frequently than sigmoid.
- **Rectified Linear Unit (ReLU).** Activation function that is zero for negative x values and a straight line for positive x values. ReLU is used more frequently than sigmoid and tanh because it's more computationally effective.
- **Tensor.** A connection between two neurons in sequential layers.
- **Cost Function (aka Loss or Objective Function).** The function that is being minimized when training the network. This function measures the difference between the desired outcome and the outcome predicted by the network. The size of this difference (as well as the step size) informs how much the parameters at each neuron are changed with each iteration.
- **Mean Squared Error.** Sum of the squared errors of each feature divided by the number of training inputs across the network.
- **Cross Entropy.** A more efficient cost function than mean squared error.
- **Dense Layer (aka Fully Connected Layer).** A layer in a neural network whose neurons connect to each of the neurons in the subsequent layer of the neural network.
- **Gradient Descent.** Methodology for figuring out how to minimize the cost function by changing weight and bias terms throughout the network
- **Learning Rate.** The speed at which the model changes weights and bias terms with each iteration. By increasing the learning rate, one increases the speed at which a model will learn but also increase the risk that the global minimum will not be found (i.e., the risk that you oscillate on either side of the global minimum because the step size is too large)
- **Weight.** Each neuron has weights that multiply each input (i.e. $w_1x_1 + w_2x_2 + b$) which goes into the activation function.
- **Bias.** Constant added to each input that is used for a neuron's activation function
- **Initialization.** The initial weights and biases that are used to calculate the outputs of each neuron in the network.
- **Softmax.** Softmax is typically used as the output layer activation function for classification. It is a proxy for probability, the output should be a proportion that approximates the probability of being a certain class, and all of the outputs should sum to 1.
- **Types of Machine Learning**
    - Reinforced Learning

  The process of teaching a machine to make specific decisions using trial and error.

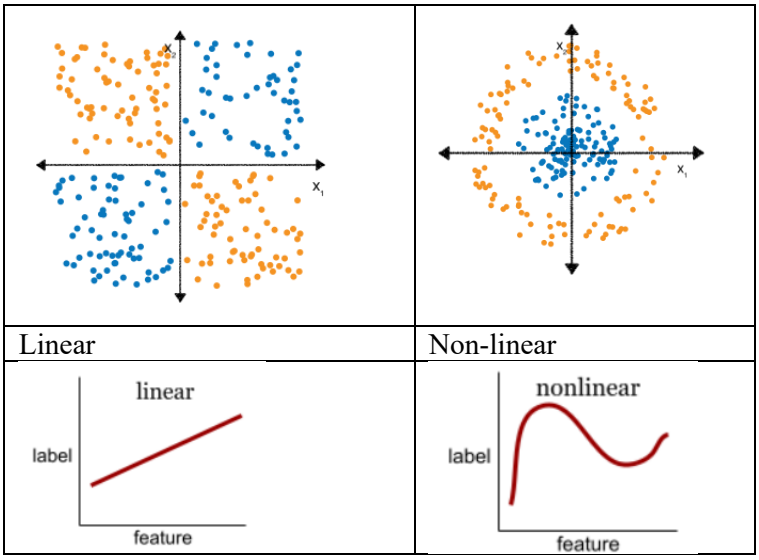
- Unsupervised Learning

Users have to look at the data and then divide it based on its own algorithms without having any training. There is no target or outcome variable to predict nor estimate.

- Supervised Learning

Users have a lot of data and can train your models. Supervised learning further falls into two groups: classification and regression.

- **Linear and non-linear classifiers.**



A linear function cannot cleanly separate all the blue dots from the orange dots. Nonlinear means that you can't accurately predict a label with a model of the form. In other words, the "decision surface" is not a line.

Assignment:

a) Design and implement a neural network that acts as a AND classifier for binary input
b) Design and implement a neural network that acts as a OR classifier for binary input
c) Design and implement a neural network that acts as a NAND classifier for binary input
d) Design and implement a neural network that acts as a XOR classifier for binary input, Comment on the inability of this NN to classify the i/p data
e) Design and implement a sequential, dense neural network that acts as a classifier for the Iris dataset, tune your neural network with different hyperparameter values for learning rate, architecture and epochs
f) Design and implement a sequential, dense neural network that acts as a classifier for the diabetes dataset provided to you in class, tune your neural network with different hyperparameter values for learning rate, architecture and epochs.
g) Design and implement a sequential, dense neural network that acts as a classifier for the heart dataset provided in class, tune your neural network with different hyperparameter values for learning rate, architecture and epochs.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well done [ ]

Signature of the instructor and date

Assignment 5: Find-S and CEA Implementation

Find-S Algorithm
The S algorithm, also known as the Find-S algorithm, is a machine learning algorithm that seeks to find a maximally specific hypothesis based on labeled training data. It starts with the most specific hypothesis and generalizes it by incorporating positive examples. It ignores negative examples during the learning process.
The algorithm's objective is to discover a hypothesis that accurately represents the target concept by progressively expanding the hypothesis space until it covers all positive instances.

Representations used by Find S Algorithm are:

- **? (Question Mark):**
  This symbol denotes a wildcard or placeholder, signifying that any value is acceptable for the corresponding attribute. Within the context of hypothesis representation, the ? allows for flexibility in capturing patterns across diverse data instances.
- **Specific Value:**
  In contrast to the wildcard represented by ?, specific attribute values are explicitly defined within a hypothesis. For instance, if a particular attribute is known to have a specific value (e.g., "Cold" for temperature), it is directly incorporated into the hypothesis.
- **ϕ (Phi):**
  The symbol ϕ represents a null or empty value, indicating that no value is acceptable for the corresponding attribute. This representation serves to delineate constraints within the hypothesis, narrowing down the range of possible attribute values.
- **Most General Hypothesis:**
  Denoted by {?, ?, ?, ?, ?, ?}, the most general hypothesis encompasses all possible attribute values, offering a broad yet inclusive representation of the concept being learned. It serves as the starting point for hypothesis refinement within the Find S Algorithm.
- **Most Specific Hypothesis:**
  In stark contrast, the most specific hypothesis is represented by {ϕ, ϕ, ϕ, ϕ, ϕ, ϕ}, where each attribute is constrained to null values. This hypothesis encapsulates the utmost specificity, essentially representing a lack of knowledge about the underlying concept.
- Hypothesis (h) : The variable h represents the hypothesis, which is the learned concept or generalization based on the training data. It is refined iteratively throughout the algorithm.

Steps of the Find-S Algorithm:

Step 1: Start with the most specific hypothesis possible (usually the null hypothesis, which does not classify any example).
Step 2: For each positive example in the training data:
If the example matches the current hypothesis, continue.
If it doesn't match, update the hypothesis to the least specific generalization that still fits the new example.
Step 3: Repeat the process for all positive examples.
Step 4: The final hypothesis will be the most specific hypothesis that fits all the positive examples.

CEA:
The CEA is a supervised machine learning algorithm used for concept learning. It aims to find a hypothesis that accurately describes a target concept based on a set of training examples. The algorithm operates by maintaining a "version space," which is the set of all

hypotheses consistent with the observed training data.

CANDIDATE-ELIMINATION Learning Algorithm

The CANDIDATE-ELIMINTION algorithm computes the version space containing all hypotheses from H that are consistent with an observed sequence of training examples.

1. Initialize G to the set of maximally general hypotheses in H
2. Initialize S to the set of maximally specific hypotheses in H
3. For each training example d, do

   If d is a positive example

   a) Remove from G any hypothesis inconsistent with d
   b) For each hypothesis s in S that is not consistent with d
   c) Remove s from S
   d) Add to S all minimal generalizations h of s such that h is consistent with d, and some member of G is more general than h
   e) Remove from S any hypothesis that is more general than another hypothesis in S

   If d is a negative example

   a) Remove from S any hypothesis inconsistent with d
   b) For each hypothesis g in G that is not consistent with d
   c) Remove g from G
   d) Add to G all minimal specializations h of g such that h is consistent with d, and some member of S is more specific than h

   Remove from G any hypothesis that is less general than another hypothesis in G

Key characteristics of CEA:

1. Incremental Learning: It learns by processing examples one at a time.
2. Guaranteed Convergence (under certain conditions): If the target concept exists within the hypothesis space and the training data is consistent, CEA will converge to the true target concept.
3. *Version Space*: The set of all hypotheses consistent with the training data.

Limitations:

Can be computationally expensive for large datasets and may struggle with noisy data or if the target concept is not representable within the chosen hypothesis space (biased hypothesis space).

Assignments:

a) Implement the Find-S algorithm on the data sets provided to you to induce hypotheses from training data

b) Implement the Candidate Elimination (aka List-Then-Eliminate) algorithm on the data sets provided to you to list of all possible hypotheses and eliminating the ones that do not fit the training examples.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well done [ ]

Signature of the instructor and date

Assignment 6: Implementation of the Gaussian Naïve Bayes Classifier

The Bayes Optimal Classifier is a probabilistic model that makes the most probable prediction for a new example.

It is described using the Bayes Theorem that provides a principled way for calculating a conditional probability. It is also closely related to the Maximum a Posteriori: a probabilistic framework referred to as MAP that finds the most probable hypothesis for a training dataset.

In practice, the Bayes Optimal Classifier is computationally expensive, if not intractable to calculate, and instead, simplifications such as the Gibbs algorithm and Naive Bayes can be used to approximate the outcome.

Bayes Theorem (Conditional Probability)
The theorem can be mathematically expressed as:

$$P(A|B)= [P(B|A)\cdot P(A)]/ P(B)$$

OR

$$P(A|B)=P(B)P(B|A)\cdot P(A)$$

where

- P(A|B) is the posterior probability of event A given event B.
- (B|A) is the likelihood of event B given event A.
- P(A) is the prior probability of event A.
- P(B) is the total probability of event B.

Assignment:

1. Design and implement the naïve Bayes classifier using the data set available at /kaggle/input/adult-dataset/adult.csv that has 15 attributes, segregate the dataset into categorical and numerical variables. Income is the target variable.
   a) Check for missing values, output a frequency count of the categorical variables and view frequency distribution of categorical variables.
   b) Check for missing values in workclass, occupation and native_country (replace ? with NaN).
   c) Check labels in workclass variable, check frequency distribution of values in workclass variable. Do the same for other two variables of (c)
   d) Print categorical variables with missing data and impute missing categorical variables with most frequent value
   e) Explore the numerical variables and problems in them (is null, sum etc)
   f) Declare the feature variables and target variable.
   g) Split the data set for train and test purpose, and make sure there are no missing values
   h) Use one-hot encoding to encode 'workclass', 'education', 'marital_status', 'occupation', 'relationship', 'race', 'sex', 'native_country'
   i) Do a feature Scaling- use RobustScaler form sklearn to transform the training and testing features (learning and test data)
   j) Fit the GaussianNB model to the training data
   k) Use the above to predict the income for the test data.
   l) Print model accuracy
   m) Check for over fitting and under fitting
   n) Compare model accuracy with null accuracy to find out how good the NB model was.
   o) Also print the confusion matrix to show number of correct predictions and incorrect

ones.

p) Print classification report using classification_report from sklearn.metrics for precision, recall, f1 and support

2. Design and implement a Multinomial Naive Bayes Classifier to classify documents into pre-defined types based on likelihood of a word occurring by using Bayes theorem. The data set shall be provided as a CSV. The dataset will be of text data categorized into four labels: **Technology**, **Sports**, **Politics** and **Entertainment**. Each entry contains a short sentence or statement related to a specific topic with the label indicating the category it belongs to.

3. Implement the NB optimal classifier. Data set is given to you in class.

Assignment Evaluation

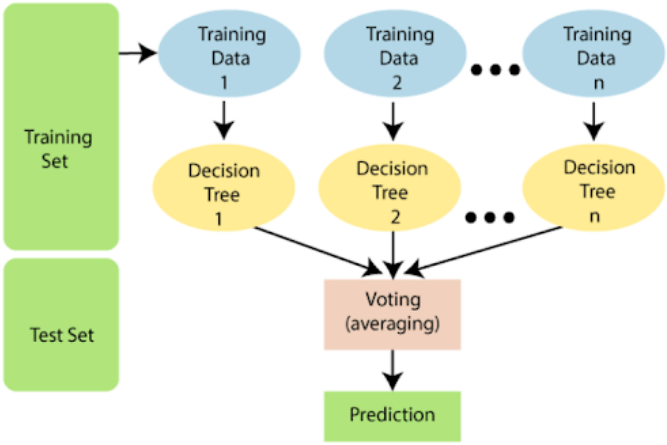
0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well done [ ]

Signature of the instructor and date

Assignment 7: Random Forest Algorithm

A Random Forest Algorithm is a supervised machine learning algorithm that is extremely popular and is used for Classification and Regression problems in Machine Learning. A forest comprises numerous trees, and the more trees more it will be robust. Similarly, the greater the number of trees in a Random Forest Algorithm, the higher its accuracy and problem-solving ability. Random Forest is a classifier that contains several decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. It is based on the concept of ***ensemble learning*** which is a process of combining multiple classifiers to solve a complex problem and improve the performance of the model.

Working of Random Forest Algorithm

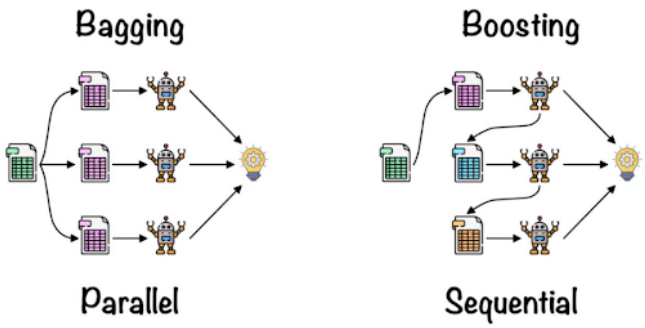


The working Random Forest Algorithm:

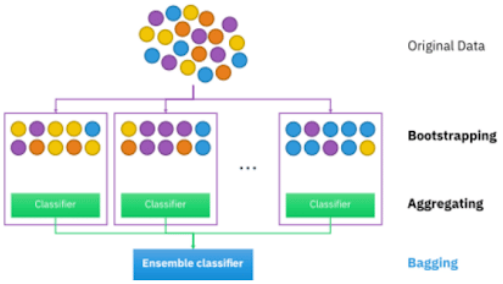
Step 1: Select random samples from a given data or training set.
Step 2: This algorithm will construct a decision tree for every training data.
Step 3: Voting will take place by averaging the decision tree.
Step 4: Finally, select the most voted prediction result as the final prediction result.

This combination of multiple models is called Ensemble. Ensemble uses two methods:

1. **Bagging**: Creating a different training subset from sample training data with replacement is called Bagging. The final output is based on majority voting.
2. **Boosting:** Combing weak learners into strong learners by creating sequential models such that the final model has the highest accuracy is called Boosting. Example: ADA BOOST, XG BOOST.



Bagging: Bagging is also known as Bootstrap Aggregation used by random forest. The process begins with any original random data. After arranging, it is organised into samples known as Bootstrap Sample. This process is known as Bootstrapping. Further, the models are trained individually, yielding different results known as Aggregation. In the last step, all the results are combined, and the generated output is based on majority voting. This step is known as Bagging and is done using an Ensemble Classifier.



Important Hyperparameters

Hyperparameters are used in random forests to either enhance the performance and predictive power of models or to make the model faster.

The following hyperparameters are used to enhance the predictive power:

- n_estimators: Number of trees built by the algorithm before averaging the products.
- max_features: Maximum number of features random forest uses before considering splitting a node.
- mini_sample_leaf: Determines the minimum number of leaves required to split an internal node.

The following hyperparameters are used to increase the speed of the model:

- n_jobs: Conveys to the engine how many processors are allowed to use. If the value is 1, it can use only one processor, but if the value is -1,, there is no limit.
- random_state: Controls randomness of the sample. The model will always produce the same results if it has a definite value of random state and if it has been given the same hyperparameters and the same training data.
- oob_score: OOB (Out Of the Bag) is a random forest cross-validation method. In this, one-third of the sample is not used to train the data but to evaluate its performance.

Assignment:

1. Implement Random Forest algorithm on MNIST data
2. Implement Random Forest algorithm on Mental health data (.csv provided in the lecture)
3. Implement Random Forest algorithm on customers' default payments data set. Given a set of features, predict the probability of a customer defaulting on a loan. The target variable is "default payment" (Yes=1; No=1)

   (download from here https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients?utm_source=ibm_developer&utm_content=in_content_link&utm_id=tutorials_awb-random-forest-predict-credit-defaults).

   a) Import the required libraries

b) Load the data set.
c) Explore the data set – drop the ID column
d) Analyze missing data - check for null values or other invalid input (use unique(), isnull(), len()- count missing data), comment on the values in EDUCATION and MARRIAGE columns, filter the rows where the EDUCATION and MARRIAGE columns have non-zero values
e) check whether the target variable is balanced using a plot
f) down sample the data - split the data based on those who defaulted on their loan and those who did not default on their loan by randomly selecting 1,000 samples from each category, and use resample()
g) Hot encode the independent variables – isolate (drop() with copy()) independent variables, encode the data using get_dummies()
h) Split the data set (downsampled data set)– train and test
i) Classify accounts and evaluate the model – print accuracy and confusion matrix
j) Optimize the model with hyperparameter tuning - n_estimators, max_depth, min_samples_split, min_samples_leaf, and max_leaf_nodes, display the confusion matrix

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well done [ ]

Signature of the instructor and date

Assignment 8: k-Nearest Neighbor

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. Mainly used for classification predictive problems in industry.
The main idea behind KNN is to find the k-nearest data points to a given test data point and use these nearest neighbors to make a prediction. The value of k is a hyperparameter that needs to be tuned, and it represents the number of neighbors to consider.
The distance metric used to measure the similarity between two data points is an essential factor that affects the KNN algorithm's performance.
The most commonly used distance metrics are Euclidean distance, Manhattan distance and Minkowski distance.

Key features of k-NN:

1. Lazy learning algorithm – k-NN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification.
   Eager learners mean when giving training data points, we will construct a generalized model before performing prediction on given new points to classify. We can think of such learners as being ready, active and eager to classify new data points.
   Lazy learning means there is no need for learning or training of the model and all of the data points are used at the time of prediction. Lazy learners wait until the last minute before classifying any data point. They merely store the training dataset and waits until classification needs to be performed. Lazy learners are also known as instance-based learners because lazy learners store the training points or instances, and all learning is based on instances.
2. Non-parametric learning algorithm − KNN is also a non-parametric learning algorithm because it doesn't assume anything about the underlying data.

The k-NN algorithm:

k-nearest neighbors (k-NN) algorithm uses 'feature similarity' to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. We can understand its working with the help of following steps −

Step 1− Load the data, split the data into training and test sets. The training set is used to train the KNN algorithm, while the test set is used to evaluate its performance.
Step 2 − Normalize the data − Before training the K-NN algorithm, it is essential to normalize the data to ensure that each feature contributes equally to the distance metric calculation.
Step 3 − choose the value of K i.e. the nearest data points. K can be any integer.
Step 4 − For each point in the test data do the following −
- 4.1 − Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.
- 4.2 − based on the distance value, sort them in ascending order.
- 4.3 − choose the top K rows from the sorted array.
- 4.4 − assign a class to the test point based on most frequent class of these rows.

Step 5 – Evaluate performance – k-NN algorithm's performance is evaluated using various metrics such as accuracy, precision, recall, and F1-score.

Pros and Cons of K-NN

Pros

- It is very simple algorithm to understand and interpret.
- It is very useful for nonlinear data because there is no assumption about data in this algorithm.
- It is a versatile algorithm as we can use it for classification as well as regression.
- It has relatively high accuracy but there are much better supervised learning models than K-NN.

Cons

- It is computationally a bit expensive algorithm because it stores all the training data.
- High memory storage required as compared to other supervised learning algorithms.
- Prediction is slow in case of big N.
- It is very sensitive to the scale of data as well as irrelevant features.

Assignment:

1. Predict Sugar of Diabetic Patient given BMI and Age using k-NN, assume k = 3

BMI	Age	Sugar
33.6	50	1
26.6	30	O
23.4	40	O
43.1	67	O
35.3	23	1
35.9	67	1
36.7	45	1
25.7	46	O
23.3	29	O
31	56	1

2. Consider the following data set

Brightness	Saturation	Class
40	20	Red
50	50	Blue
60	90	Blue
10	25	Red
70	70	Blue
60	10	Red
25	80	Blue

Design a k-NN to assign a class label for the following.

Brightness	Saturation	Class
20	35	?

3. Apply k-NN on the iris data set

Assignment Evaluation

0: Not Done [ ] 1: Incomplete [ ] 2: Late Complete [ ]
3: Needs Improvement [ ] 4: Complete [ ] 5: Well done [ ]

Signature of the instructor and date