Experiment 11 - Analysis and comparison of different Machine learning and Deep learning algorithms

Roll No.	37
Name	Mikil Lalwani
Class	D20 B
Subject	Data Science Lab
LO Mapped	L4: Develop real life applications using learning concepts. L5: Evaluate performance of applications.

Aim: Analysis and comparison of different Machine learning and Deep learning algorithms

Theory:

Machine learning:

K-Nearest Neighbor(KNN) Algorithm for Machine Learning

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know whether it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

Deep Learning:

Feed Forward Process in Deep Neural Network

Multilayer Feed-Forward Neural Network(MFFNN) is an interconnected Artificial Neural Network with multiple layers that has neurons with weights associated with them and they compute the result using activation functions. It is one of the types of Neural Networks in which the flow of the network is from input to output units and it does not have any loops, no feedback, and no signal moves in backward directions that is from output to hidden and input layer.

The ANN is a self-learning network that learns from sample data sets and signals, it is based on the function of the biological nervous system. The type of activation function depends on the desired output. It is a part of machine learning and AI, which are the fastest-growing fields, and lots of research is going on to make it more effective.

The Architecture of the Multilayer Feed-Forward Neural Network:

This Neural Network or Artificial Neural Network has multiple hidden layers that make it a multilayer neural Network and it is feed-forward because it is a network that follows a top-down approach to train the network. In this network there are the following layers:

1. **Input Layer:** It is the starting layer of the network that has a weight associated with the signals.
2. **Hidden Layer:** This layer lies after the input layer and contains multiple neurons that perform all computations and pass the result to the output unit.
3. **Output Layer:** It is a layer that contains output units or neurons and receives processed data from the hidden layer, if there are further hidden layers connected to it then it passes the weighted unit to the connected hidden layer for further processing to get the desired result.

The input and hidden layers use sigmoid and linear activation functions whereas the output layer uses a Heaviside step activation function at nodes because it is a two-step activation function that helps in predicting results as per requirements. All units also known as neurons have weights and calculation at the hidden layer is the summation of the dot product of all weights and their signals and finally the sigmoid function of the calculated sum. Multiple hidden and output layers increases the accuracy of the output.

Python Library Function Used:

Topic: Lung Cancer Detection.
Dataset: https://www.kaggle.com/datasets/mysarahmadbhat/lung-cancer

1. **pandas -**
   Pandas is an open-source library in Python that is made mainly for working with relational or labelled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library of Python. Pandas is fast and it has high performance & productivity for users.

2. **Seaborn -**
   Seaborn is a Python data visualisation library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

3. **LabelEncoder -**
   LabelEncoder is a utility class from the scikit-learn (sklearn) library in Python used for encoding categorical labels into numerical values. It assigns a unique integer to each distinct label in a categorical variable, effectively converting them into a format that machine learning algorithms can work with. This transformation is particularly useful when dealing with classification tasks where the target variable or class labels need to be represented numerically. LabelEncoder is a simple and commonly used tool for this purpose in scikit-learn.

4. **KNN Classification -**

   The KNeighborsClassifier is a machine learning algorithm provided by scikit-learn (sklearn) used for both classification and regression tasks. Here's a short description:

   The KNeighborsClassifier is a type of instance-based or memory-based supervised learning algorithm. It operates on the principle of finding the K-nearest data points in the training dataset to a given input data point and making predictions based on the majority class (for classification) or averaging (for regression) among those K neighbors

5. **MLPClassifier (Deep learning algorithm) -**

The MLPClassifier is a machine learning algorithm from scikit-learn (sklearn) that represents a Multi-layer Perceptron (MLP) for classification tasks. It's a feedforward neural network consisting of input, hidden, and output layers, with configurable architecture and activation functions. The MLPClassifier is used for supervised learning, specifically for classification problems. It can handle complex relationships in the data and has the flexibility to accommodate various architectures, making it suitable for a wide range of classification tasks.

Code and Observation

1. Importing dataset

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
```

```
[ ] df = pd.read_csv("sample_data/survey lung cancer.csv")
```

```
[ ] df.head()
```

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	ALLERGY
0	M	69	1	2	2	1	1	2	1
1	M	74	2	1	1	1	2	2	2
2	F	59	1	1	1	2	1	2	1
3	M	63	2	2	2	1	1	1	1
4	F	63	1	2	1	1	1	1	1

2. Cleaning dataset

Drop Duplicates Values

```
[ ] df.drop_duplicates()
```

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE
0	M	69	1	2	2	1
1	M	74	2	1	1	1
2	F	59	1	1	1	2
3	M	63	2	2	2	1
4	F	63	1	2	1	1
...	...	...	...	...	...	...
279	F	59	1	2	2	2
280	F	59	2	1	1	1
281	M	55	2	1	1	1
282	M	46	1	2	2	1
283	M	60	1	2	2	1

276 rows × 16 columns

3. Label encoding string values into numerical values.

```
print(df['GENDER'].unique())
print(df['LUNG_CANCER'].unique())
```

```
['M' 'F']
['YES' 'NO']
```

```
[ ] le = LabelEncoder()
    df['GENDER'] = le.fit_transform(df['GENDER'])
    df['LUNG_CANCER'] = le.fit_transform(df['LUNG_CANCER'])
```

```
[ ] df.head()
```

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	AL
0	1	69	1	2	2	1	1	2	
1	1	74	2	1	1	1	2	2	
2	0	59	1	1	1	2	1	2	
3	1	63	2	2	2	1	1	1	
4	0	63	1	2	1	1	1	1	

GENDER: 1 - Male, 0 - Female LUNG_CANCER: 1 - YES, 0 - NO

4. Splitting dataset into train and test set.

```
[ ] X = df.drop('LUNG_CANCER', axis=1)
    Y = df['LUNG_CANCER']
```

```
[ ] from sklearn.model_selection import train_test_split

    X_train, X_test, Y_train, Y_test = train_test_split(X, Y,test_size=0.2, random_state=7)
```

Supervised Learning - KNN Classifier

5. Creating KNN classification model.

```
0s
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, Y_train)

knn_pred = knn.predict(X_test)
print(knn_pred)

knn_accuracy = accuracy_score(Y_test, knn_pred)
print('Accuracy: ',knn_accuracy)
```

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1]
Accuracy:  0.8360655737704918
```

Deep Learning - Feed Forward Neural Network

6. Creating a Feed Forward Neural network.

```python
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, classification_report

mlp_classifier = MLPClassifier(hidden_layer_sizes=(128, 64), activation='relu', max_iter=1000, random_state=42)

mlp_classifier.fit(X_train, Y_train)


y_pred = mlp_classifier.predict(X_test)


accuracy = accuracy_score(Y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')


classification_rep = classification_report(Y_test, y_pred)
print('\nClassification Report:\n', classification_rep)
```

```
Accuracy: 0.85

Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         7
           1       0.88      0.96      0.92        54

    accuracy                           0.85        61
   macro avg       0.44      0.48      0.46        61
weighted avg       0.78      0.85      0.81        61
```

Conclusion:

Deep learning algorithm - Feed Forward Neural Network proved to be more accurate than the machine learning algorithm KNN classification on the Lung Cancer Detection dataset.

Model	Accuracy
Feed Forward Neural Network	0.85
KNN Classifier	0.83