Training Report – Day 10

Topic Covered Today:

- Understanding **KNN (K-Nearest Neighbors)** Algorithm
- Introduction to **CNN (Convolutional Neural Networks)**
- Overview of **LAMP Stack** and its components

Key Learning:

KNN (K-Nearest Neighbors):

KNN is a **supervised machine learning algorithm** used for **classification and regression** tasks. It works by comparing the distance between data points and predicting the output based on the majority of its nearest neighbors.

Key Concepts:

- It stores all available data and classifies new data points based on similarity (distance).
- Common distance metrics: **Euclidean**, **Manhattan**, **Minkowski**.
- The value of **K** determines how many neighbors are considered.

Example (in Python):

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

data = load_iris()
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, test_size=0.2)

model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)
pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, pred))
```

Output: The model predicts flower species based on features like petal length and width, giving around **95% accuracy** for K=3.

CNN (Convolutional Neural Networks):

CNN is a **deep learning algorithm** mainly used for **image classification and recognition**. It automatically detects important features like edges, colors, and shapes from images without manual feature extraction.

Key Layers of CNN:

1. **Convolution Layer** – Extracts features using filters (kernels).
2. **Pooling Layer** – Reduces spatial dimensions and computation.
3. **Fully Connected Layer** – Connects neurons for final classification.

Applications:

- Face recognition
- Object detection
- Medical image analysis
- Emotion and facial expression detection

Example (simple CNN structure):

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(64,64,3)),
    MaxPooling2D(pool_size=(2,2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
```

This structure defines a CNN that can be trained on image datasets like MNIST or CIFAR.

LAMP Stack:

LAMP stands for **Linux, Apache, MySQL, and PHP/Python/Perl** — a combination of technologies used for web development and deployment.

Components:

- **Linux:** The operating system base.
- **Apache:** Web server that handles HTTP requests.
- **MySQL:** Database system for storing web data.
- **PHP/Python/Perl:** Backend scripting languages for web applications.

Usage in AI/ML Projects:

- LAMP can host dashboards or APIs for AI/ML models.
- Helps deploy trained models as web-based applications.

Activities / Assignments:

- Implemented **KNN algorithm** on the Iris dataset using Scikit-learn.
- Understood the architecture of **CNN** and its real-world applications.
- Installed and explored **LAMP Stack** environment on Ubuntu Linux.
- Connected a sample web page to a **MySQL database** using PHP.
- Noted down differences between **traditional ML (KNN)** and **deep learning (CNN)**.

Personal Reflection for Day 10:

Today's session was very insightful as it combined concepts of **machine learning, deep learning, and web development**. I learned how KNN works on small structured datasets and how CNNs are used for complex image-based problems.

Understanding the **LAMP stack** helped me realize how backend environments can be used to deploy AI/ML models as real-world web applications. It was exciting to see how these technologies connect together — from model training (ML/DL) to deployment (LAMP).

This session gave me a complete overview of how data science models can move from experimentation to real-world use.