Training Report – Day 14

Topic Covered Today:

- Introduction to **Logistic Regression**
- Difference between **Linear and Logistic Regression**
- Mathematical concept and **Sigmoid Function**
- Implementation of Logistic Regression using **Python (Scikit-learn)**
- Model evaluation using accuracy and confusion matrix

Key Learning:

Introduction to Logistic Regression:

Today I learned about **Logistic Regression**, a type of **supervised learning algorithm** used for **classification problems**.
Unlike Linear Regression (which predicts continuous values), Logistic Regression predicts **categorical outcomes** — for example, whether an email is spam or not, or whether a tumor is benign or malignant.

Logistic Regression predicts the probability that a given input belongs to a particular class (0 or 1).

Mathematical Concept:

The main idea behind Logistic Regression is the **Sigmoid Function**, also known as the **Logistic Function**, which converts any real number into a probability value between 0 and 1.

If the output of $(h(x)) > 0.5 \rightarrow$ class = 1
If the output of $(h(x)) \leq 0.5 \rightarrow$ class = 0

This makes it ideal for **binary classification** problems.

Difference between Linear and Logistic Regression:

Aspect	Linear Regression	Logistic Regression

Aspect	Linear Regression	Logistic Regression
Output Type	Continuous (real values)	Categorical (0 or 1)
Equation	(Y = mX + c)	(Y = \frac{1}{1 + e^{-(mX + c)}})
Use Case	Predicting numbers (e.g., price, temperature)	Classification (e.g., spam or not spam)
Loss Function	Mean Squared Error	Log Loss / Binary Cross Entropy

---

Implementation in Python:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

# Example dataset
X = [[1], [2], [3], [4], [5], [6]]
y = [0, 0, 0, 1, 1, 1]

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Create and train model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
print("Predictions:", y_pred)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

Output Example:

```
Predictions: [0 1]
Accuracy: 1.0
Confusion Matrix:
[[1 0]
 [0 1]]
```

This shows that the model classified the samples correctly with 100% accuracy on this simple dataset.

Evaluation Metrics:

To assess the performance of a Logistic Regression model, I learned about:

- **Accuracy** → (Correct Predictions / Total Predictions) × 100
- **Confusion Matrix** → shows True Positives, True Negatives, False Positives, False Negatives
- **Precision, Recall, F1-score** → important for unbalanced datasets

---

Activities / Assignments:

- Understood the concept and mathematical background of **Logistic Regression**.
- Practiced implementing the algorithm using **Scikit-learn**.
- Compared Linear and Logistic Regression outputs.
- Evaluated model performance using **accuracy** and **confusion matrix**.
- Visualized the **Sigmoid Curve** using Matplotlib to understand probability mapping.

Personal Reflection for Day 14:

Today's session helped me understand how **classification models** work and how logistic regression forms the foundation for many AI applications like medical diagnosis, email filtering, and fraud detection.

I realized that although it's called "regression," Logistic Regression is mainly used for **classification tasks**. The concept of the **sigmoid function** and probability-based decision-making was very interesting.

Implementing the model in Python made it easier to understand how algorithms convert numerical features into class predictions. This session strengthened my understanding of core ML algorithms and their practical use in real-world data classification problems.