Training Report – Day 18

Topic Covered Today:

- Introduction to **Sentiment Analysis**
- Text preprocessing and feature extraction
- Building a sentiment analysis model using **Python and NLP**
- Applications of sentiment analysis in real life

Key Learning:

Introduction to Sentiment Analysis:

Today I learned about **Sentiment Analysis**, which is a popular application of **Natural Language Processing (NLP)**. It helps determine whether a given piece of text expresses a **positive, negative, or neutral emotion**.

It is widely used in **social media monitoring, customer feedback analysis, brand reputation tracking, and product review classification**.

In simple terms, sentiment analysis helps computers understand **human emotions from text data**.

Text Preprocessing Steps:

Before performing sentiment analysis, I learned to clean and prepare the text data:

1. **Tokenization:** Splitting sentences into individual words.
2. **Lowercasing:** Converting all words to lowercase.
3. **Removing Stopwords:** Eliminating common words (e.g., "is," "the," "and").
4. **Removing Punctuation:** Cleaning unnecessary symbols.
5. **Stemming or Lemmatization:** Reducing words to their base form (e.g., "liked" → "like").

These steps ensure the text is ready for machine learning algorithms to process effectively.

Feature Extraction:

To make text understandable by a model, it needs to be converted into **numerical format**. Common techniques used:

- **Bag of Words (BoW)**
- **TF-IDF (Term Frequency–Inverse Document Frequency)**
- **Word Embeddings (Word2Vec, BERT)**

Implementation in Python:

I practiced a **simple sentiment analysis model** using sklearn and nltk.

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB

# Sample dataset
texts = ["I love this product", "This is terrible", "Amazing quality", "I hate this", "It's okay, not great"]
labels = [1, 0, 1, 0, 1]  # 1 = Positive, 0 = Negative

# Convert text to numeric data
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(texts)

# Train model
model = MultinomialNB()
model.fit(X, labels)

# Test prediction
test = ["I really love this phone"]
test_vector = vectorizer.transform(test)
print("Prediction:", model.predict(test_vector))
```

Output Example:

```
Prediction: [1]
```

✅The model predicted a **positive sentiment** for the input sentence "I really love this phone."

Evaluation Metrics:

To evaluate sentiment models, I learned about key performance metrics:

- **Accuracy** – Percentage of correct predictions.
- **Precision** – How many predicted positives were actually positive.
- **Recall** – How many actual positives were correctly predicted.

- **F1 Score** – Harmonic mean of precision and recall.

Applications of Sentiment Analysis:

- **Social Media Monitoring** – Tracking opinions about brands or events.
- **Product Review Analysis** – Understanding customer satisfaction.
- **Political Analysis** – Analyzing public opinion on policies or candidates.
- **Customer Support** – Identifying emotional tone in customer feedback.
- **Market Research** – Understanding trends through people's sentiments.

Activities / Assignments:

- Collected sample text data and labeled it manually as positive or negative.
- Preprocessed text using **NLTK functions** (tokenization, stopword removal, lemmatization).
- Built and trained a **Naive Bayes classifier** for sentiment prediction.
- Evaluated accuracy and tested the model on new sentences.
- Compared outputs between **Bag of Words** and **TF-IDF** representations.

Personal Reflection for Day 18:

Today's session was very engaging because it showed how machines can analyze human emotions and opinions from text. I enjoyed applying the concepts of **NLP** from the previous day to a practical real-world task.

It was interesting to see how a few lines of Python code can determine whether a sentence expresses positive or negative sentiment. This made me realize how AI is used behind the scenes on platforms like Twitter, Amazon, and YouTube to analyze comments and reviews.

Overall, this session helped me understand how **data preprocessing, feature extraction, and machine learning models** come together to perform powerful text analysis tasks.