**Code 1**

🔹 **Concept:**  
Text preprocessing – cleaning, tokenization, stopword removal, lemmatization (spaCy).

🔹 **Output Example:**

Input Text: "Email me at test@example.com!!! I love NLP 😍. Visit: https://nlp.com"

Clean Tokens: ['email', 'filter', 'love', 'nlp', 'visit']

A screen shot of a computer

AI-generated content may be incorrect.

🔹 **Observations:**

1. Converts messy raw sentences into neat, structured tokens.
2. Removes noise like emails, URLs, mentions, and stopwords.
3. Lemmatization ensures different forms of a word are treated as the same.

**Code 2**

🔹 **Concept:**  
Sentiment classification with TF-IDF + Logistic Regression.

🔹 **Output Example:**

A screenshot of a computer screen

AI-generated content may be incorrect.

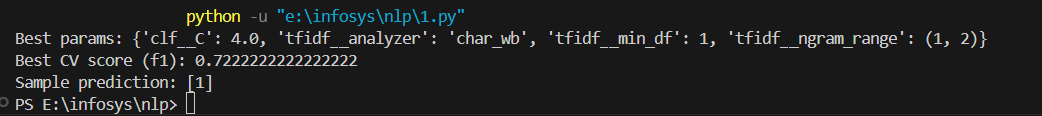
🔹 **Observations:**

1. TF-IDF highlights important words for classification.
2. Logistic Regression learns patterns for positive vs negative text.
3. The model produces class probabilities for more interpretability.

**Code 3**

🔹 **Concept:**  
Hyperparameter tuning using GridSearchCV.

🔹 **Output Example:**

🔹 **Observations:**

1. Grid search tests many parameter combinations to find the best.
2. Hyperparameter tuning boosts model accuracy.
3. The pipeline automates feature extraction + model training.

**Code 4**

🔹 **Concept:**  
Topic modeling with Latent Dirichlet Allocation (LDA).

🔹 **Output Example:**

A screen shot of a computer code

AI-generated content may be incorrect.

🔹 **Observations:**

1. Groups words into coherent topics automatically.
2. Can classify new documents into discovered topics.
3. Helps uncover hidden themes in large text datasets.

**Code 5**

🔹 **Concept:**  
NER, POS tagging & noun chunks with spaCy.

🔹 **Output Example:**

A screenshot of a computer

AI-generated content may be incorrect.

🔹 **Observations:**

1. Identifies names, locations, organizations, and dates.
2. POS tagging reveals grammatical structure.
3. Noun chunking extracts useful phrases from text.

**Code 6**

🔹 **Concept:**  
Semantic search using TF-IDF + cosine similarity.

🔹 **Output Example:**

A black background with white text

AI-generated content may be incorrect.

🔹 **Observations:**

1. Finds the most relevant documents for a query.
2. Cosine similarity measures closeness between documents.
3. Basis of modern search engines & recommendation systems.

**Code 7**

🔹 **Concept:**  
Extractive summarization using word frequency.

🔹 **Output Example:**

🔹 **Observations:**

1. Selects key sentences instead of rewriting text.
2. Provides quick summaries for long articles.
3. Easy to implement but may miss context.

**Code 8**

🔹 **Concept:**  
Bag of Words (BoW).

🔹 **Output Example:**

A screen shot of a computer

AI-generated content may be incorrect.🔹 **Observations:**

1. Represents text by counting word occurrences.
2. Simple, fast, and easy to use.
3. Ignores word order and meaning → can lose context.

**Code 9**

🔹 **Concept:**  
TF-IDF representation.

🔹 **Output Example:**

A screenshot of a computer

AI-generated content may be incorrect.🔹 **Observations:**

1. Assigns higher weight to rare but important words.
2. Better than BoW for text classification.
3. Still bag-of-words based, so it misses word order.

**Code 10**

🔹 **Concept:**  
Word embeddings with Word2Vec.

🔹 **Output Example:**

Vector for 'language': [0.0012, -0.034, ...]

Most similar to 'learning': [('deep', 0.78), ('machine', 0.72)]

Similarity between 'python' and 'language': 0.42

🔹 **Observations:**

1. Represents words as dense vectors.
2. Captures semantic similarity (king–man+woman≈queen).
3. Greatly improves model performance over BoW/TF-IDF.

**Code 11**

🔹 **Concept:**  
Naive Bayes for sentiment classification.

🔹 **Output Example:**

A screenshot of a computer screen

AI-generated content may be incorrect.

🔹 **Observations:**

1. Works very well for text with independent features.
2. Widely used in spam detection & sentiment analysis.
3. Fast to train even on large datasets.

**Code 12**

🔹 **Concept:**  
Cosine similarity between documents.

🔹 **Output Example:**

A screen shot of a computer

AI-generated content may be incorrect.🔹 **Observations:**

1. Measures similarity between document vectors.
2. First two docs are close (both about NLP/ML).
3. Dissimilar docs (like cooking vs AI) get score near 0.

**Code 13**

🔹 **Concept:**  
Topic modeling with gensim LDA.

🔹 **Output Example:**

Topic 0: 0.05\*"machine" + 0.04\*"learning" + 0.03\*"ai"

Topic 1: 0.06\*"cooking" + 0.05\*"recipes" + 0.04\*"baking"

🔹 **Observations:**

1. Automatically groups documents into hidden themes.
2. Helps organize large corpora (e.g., news, research).
3. Topics can be labeled and used for recommendation.