Using Pinecone to Retrieve Top 3 Most Similar Product Records

1. Objective

- Learn to initialize and interact with the Pinecone vector database using the official Python client.
- Set up a Pinecone index and upsert sample vectors representing product data.
- Perform similarity search to retrieve the top three most similar records for a given input query.

2. Problem Statement

- In e-commerce and recommendation systems, efficiently retrieving the most similar products based on descriptions or features is crucial.
- This exercise guides you to use Pinecone's vector search to store product embeddings and find the nearest neighbors for a new query embedding.

3. Inputs / Shared Artifacts

You will work with the following sample product dataset embedded within your script:

```
products = [
    {"id": "prod1", "title": "Red T-Shirt", "description": "Comfortable
cotton t-shirt in bright red", "embedding": [0.12, 0.98, 0.34, 0.56]},
    {"id": "prod2", "title": "Blue Jeans", "description": "Stylish
denim jeans with relaxed fit", "embedding": [0.10, 0.88, 0.40, 0.60]},
    {"id": "prod3", "title": "Black Leather Jacket", "description":
"Genuine leather jacket with classic style", "embedding": [0.90, 0.12,
0.75, 0.15]},
    {"id": "prod4", "title": "White Sneakers", "description":
"Comfortable sneakers perfect for daily wear", "embedding": [0.20,
0.95, 0.38, 0.55]},
    {"id": "prod5", "title": "Green Hoodie", "description": "Warm
hoodie made of organic cotton", "embedding": [0.15, 0.93, 0.35, 0.50]},
]
```

4. Expected Outcome

- Pinecone index is created and populated with the sample product embeddings.
- Given a new input embedding (e.g., representing a search query), the program returns the top three most similar products by cosine similarity.
- Proper handling of Pinecone client initialization, index creation, upserting vectors, querying, and closing the connection.

5. Concepts Covered

- Initializing Pinecone client with API key.
- Creating and configuring a Pinecone index.
- Upserting vectors into the index.
- Performing similarity search queries.
- Retrieving and interpreting top-k results.
- Managing resources and error handling with Pinecone.

6. Example: Step-by-Step Instructions with Code

```
from pinecone import Pinecone

# Step 1: Initialize Pinecone client
pc = Pinecone(api_key="<PINECONE_TOKEN>")  # Replace with your actual
Pinecone API key

# Step 2: Create or connect to an index
```

```
index_name = "product-similarity-index"
if index_name not in pc.list_indexes():
    pc.create_index(name=index_name, dimension=4)  # dimension matches
your embedding size

index = pc.Index(index_name)

# Step 3: Upsert sample product vectors into the index
products = [
    {"id": "prod1", "title": "Red T-Shirt", "embedding": [0.12, 0.98,
0.34, 0.56]},
    {"id": "prod2", "title": "Blue Jeans", "embedding": [0.10, 0.88,
0.40, 0.60]},
    {"id": "prod3", "title": "Black Leather Jacket", "embedding":
[0.90, 0.12, 0.75, 0.15]},
    {"id": "prod4", "title": "White Sneakers", "embedding": [0.20,
0.95, 0.38, 0.55]},
    {"id": "prod5", "title": "Green Hoodie", "embedding": [0.15, 0.93,
0.35, 0.50]},
]

vectors = [(p["id"], p["embedding"]) for p in products]
index.upsert(vectors)

# Step 4: Prepare input query embedding
query_embedding = [0.18, 0.90, 0.40, 0.52]  # Example embedding
representing user search input

# Step 5: Query Pinecone index for top 3 most similar vectors
top_k = 3
results = index.query(queries=[query_embedding], top_k=top_k,
include_metadata=False)

# Step 6: Display results
print(f"Top {top_k} similar products for the query:")
for match in results.matches:
    product_id = match.id
    score = match.score
```

```
        # Find product details
        product = next(p for p in products if p["id"] == product_id)
        print(f"- {product['title']} (Similarity score: {score:.4f})")

    # Step 7: Cleanup if needed
    pc.close()
```

7. Final Submission Checklist

- Submit your Python script that:
    - Initializes Pinecone client and index.
    - Upserts sample data vectors.
    - Queries the index with a sample input embedding.
    - Prints the top 3 most similar product titles and similarity scores.
- Include your sample dataset embedded in code.
- Provide console output demonstrating similarity search results in Jupyter Notebook file or README.
- (Optional) Write a short explanation of your approach and any challenges faced.