6) VSM

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
def vector_space_model(docs, query):
    vectorizer = TfidfVectorizer()
    vectors = vectorizer.fit_transform(docs + [query])
    cosine_similarities = cosine_similarity(vectors[-1], vectors[:-1]).flatten()
    ranked_docs = sorted(enumerate(cosine_similarities), key=lambda x: x[1], reverse=True)
    return [(doc_id, round(similarity, 2)) for doc_id, similarity in ranked_docs]
# Example usage
docs = ["machine learning and data mining", "information retrieval system", "deep learning applications"]
query = "data mining system"
print(vector_space_model(docs, query))
```

7) BIM

```
def boolean_ir_model(index, query):
    query_terms = query.lower().split()
    result_set = set(index.get(query_terms[0], []))
    for i in range(1, len(query_terms), 2):
        operator = query_terms[i]
        term = query_terms[i + 1]
        postings = set(index.get(term, []))
        if operator == "and":
            result_set = result_set.intersection(postings)
        elif operator == "or":
            result_set = result_set.union(postings)
    return sorted(result_set)
# Example usage
index = spimi_algorithm(["data mining is fun", "information retrieval systems", "mining and retrieval"])
print(boolean_ir_model(index, "mining and retrieval"))
```

8) SVM

```
from sklearn import svm
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
# Example data
docs = ["I love programming", "Python is great for data science", "Data mining is fun",
        "I dislike bugs in the code", "Debugging is frustrating"]
labels = [1, 1, 1, 0, 0]  # 1 for positive, 0 for negative sentiment
def svm_classification(docs, labels):
    vectorizer = TfidfVectorizer()
    X = vectorizer.fit_transform(docs)
    X_train, X_test, y_train, y_test = train_test_split(X, labels, test_size=0.2, random_state=42)
    classifier = svm.SVC(kernel='linear')
    classifier.fit(X_train, y_train)
    predictions = classifier.predict(X_test)
    accuracy = accuracy_score(y_test, predictions)
    return accuracy, predictions
# Example usage
accuracy, predictions = svm_classification(docs, labels)
print(f"Accuracy: {accuracy}")
print(f"Predictions: {predictions}")
```