```
# ---------------------------------------------------------
# Block 1: Installation, Imports, and Configuration
# Run this block once to set up the environment.
# ---------------------------------------------------------

# Install required libraries
print("Installing dependencies...")
!pip install -q faiss-cpu sentence-transformers pandas numpy

# Standard imports
import pandas as pd
import numpy as np
import faiss
import re
import os
from sentence_transformers import SentenceTransformer
import warnings

# Suppress warnings for cleaner output
warnings.filterwarnings('ignore')

# Configuration
CSV_FILENAME = 'testing1.csv'
MODEL_NAME = 'all-MiniLM-L6-v2'
K_RESULTS = 5 # Number of top results to show


CRIME_KEYWORDS = [
    'abducting', 'abduction', 'abetment', 'abetment to suicide', 'abetting', 'abetting mutiny',
    'abuse', 'acid attack', 'adminstration', 'adulteration', 'aggravated assault', 'arson',
    'arsonist', 'assault', 'attempt to murder', 'attempted murder', 'battery', 'bigamy',
    'blackmail', 'bomb', 'bombing', 'breach of contract', 'bribery', 'bribing', 'burglary',
    'causing miscarriage', 'cheating', 'child abuse', 'child pornography', 'concealment',
    'confinement', 'conspiracy', 'corruption', 'counterfeit', 'counterfeiting',
    'criminal breach of trust', 'criminal intimidation', 'criminal trespass', 'cruelty',
    'culpable homicide', 'cyber fraud', 'cybercrime', 'cyberstalking', 'dacoity', 'damage',
    'data breach', 'death by negligence', 'defamation', 'defiling', 'defiling place worship',
    'desertion', 'disappearance of evidence', 'dishonestly', 'domestic violence', 'dowry',
    'dowry death', 'drug trafficking', 'drunk driving', 'embezzlement', 'eve teasing', 'exciting',
    'extorting', 'extortion', 'fabricating false evidence', 'false charge', 'false claim',
    'false evidence', 'false personation', 'false statement', 'forgery', 'fornication',
    'fraud','gambling','grievous hurt', 'harassment', 'hijacking', 'hit and run', 'homicide', 'hostage',
    'housebreaking', 'human trafficking', 'hurt', 'identity fraud', 'identity theft', 'illegal weapon',
    'impersonation', 'imputation', 'indecent', 'intimidation', 'kidnap for ransom', 'kidnapping',
    'larceny','liquor', 'manslaughter', 'mischief', 'molestation', 'money', 'money laundering', 'murder',
    'mutilating', 'mutilation', 'mutiny', 'narcotics','narcotics possession', 'obscene', 'obstructing public servant',
    'obstruction', 'organized crime', 'perjury', 'phishing', 'piratical', 'poisoning', 'prostitution',
    'public nuisance', 'rape', 'rash driving', 'receiving', 'receiving stolen property', 'restraint',
    'rioting','ritualism', 'robbery', 'sedition', 'seducing', 'sexual assault', 'sexual harassment', 'shoplifting',
    'smuggling', 'snatching', 'stalking', 'stole', 'tampering with evidence', 'terrorism', 'theft',
    'threats', 'torture', 'trafficking', 'trespass', 'unauthorized access', 'unlawful assembly',
    'unnatural', 'uttering', 'vandalism', 'vehicle theft','voyeurism', 'violence', 'weapon', 'weapons',
    'wildlife crimes', 'wrongful', 'wrongful confinement', 'wrongful restraint'
]

print("Block 1 execution complete: Dependencies installed and configurations set.")
```

```
Installing dependencies...
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23.7/23.7 MB 48.1 MB/s eta 0:00:00
WARNING:torchao.kernel.intmm:Warning: Detected no triton, on systems without Triton certain kernels will not work
Block 1 execution complete: Dependencies installed and configurations set.
```

New Section

```
# ---------------------------------------------------------
# Block 2: Data Loading Functions and FIR Preprocessor Class
# Defines how the CSV is loaded and how FIR reports are cleaned/normalized.
# This block handles the file upload prompt.
# ---------------------------------------------------------

def ensure_data_exists(filename):
    """Checks if the CSV exists and prompts for upload in Colab if missing."""
    if os.path.exists(filename):
        print(f"File '{filename}' found.")
        return

    print(f"File '{filename}' not found.")
    try:
        from google.colab import files
        print("Please upload the CSV file (e.g., 'testing1.csv') now:")
        files.upload()
        if not os.path.exists(filename):
            print(f"ERROR: Uploaded file not named '{filename}'. Please rename or upload correctly.")
    except ImportError:
        print("Not running in Google Colab. Please ensure the CSV is in the directory.")

def load_data(path):
    """Loads the BNS dataset."""
    ensure_data_exists(path)
    try:
        df = pd.read_csv(path, encoding='latin1')
        print(f"Successfully loaded {len(df)} BNS sections.")
        return df
    except Exception as e:
        print(f"Error loading file: {e}")
        return pd.DataFrame({'Section': ['Error'], 'Description': ['File could not be loaded.']})

class FIRPreprocessor:
    """Class to normalize the raw FIR text into an optimized query."""
    def __init__(self, keywords):
        self.keywords = set(k.lower() for k in keywords)

    def normalize(self, text):
        """Identifies crime keywords and removes administrative noise."""
        text_lower = text.lower()

        # Step A: Identify Keywords
        found_keywords = [kw for kw in self.keywords if kw in text_lower]

        # Step B: Remove Noise (Dates, Times, Headers)
        text_clean = re.sub(r'\d{1,2}[/-]\d{1,2}[/-]\d{2,4}', '', text) # Dates
        text_clean = re.sub(r'\d{1,2}:\d{2}\s*(?:AM|PM|am|pm)?', '', text_clean) # Times

        noise_phrases = [
            r'to the station house officer', r'subject:', r'respeceted sir',
            r'i am writing to report', r'located at', r'a case has been registered'
        ]
        for phrase in noise_phrases:
            text_clean = re.sub(phrase, '', text_clean, flags=re.IGNORECASE)

        text_clean = re.sub(r'\s+', ' ', text_clean).strip()

        # Step C: Construct Normalized Query (Weighted with keywords)
        if found_keywords:
            unique_kws = list(set(found_keywords))
            normalized_query = f"Crime Categories: {', '.join(unique_kws)}. Context: {text_clean}"
        else:
            normalized_query = text_clean

        return normalized_query, found_keywords

print("Block 2 execution complete: Preprocessing class and data loaders defined.")
```

```
Block 2 execution complete: Preprocessing class and data loaders defined.
```

```
# ---------------------------------------------------------
# Block 3: FAISS Search Engine Class Definition and Initialization
# Loads the data, loads the Sentence-BERT model, and builds the FAISS index.
# This is the most time-consuming step and should only be run once.
# ---------------------------------------------------------

class BNSSearchEngine:
    """Manages the Sentence-BERT model and the FAISS index."""
    def __init__(self, df, model_name):
        self.df = df
        print("\n[STEP 1/3] Loading Sentence-BERT model...")
        self.model = SentenceTransformer(model_name)
        self.index = None
        self._build_index()

    def _build_index(self):
        print("[STEP 2/3] Generating embeddings for BNS descriptions...")
        descriptions = self.df['Description'].fillna('').tolist()
        self.embeddings = self.model.encode(descriptions)

        dimension = self.embeddings.shape[1]
        self.index = faiss.IndexFlatL2(dimension)
        self.index.add(self.embeddings.astype(np.float32))
        print(f"[STEP 3/3] FAISS Index built with {self.index.ntotal} vectors of dimension {dimension}.")

    def search(self, query_text, k):
        """Encodes query and searches FAISS index."""
        query_vec = self.model.encode([query_text]).astype(np.float32)
        distances, indices = self.index.search(query_vec, k)

        results = []
        for i, idx in enumerate(indices[0]):
            if idx < len(self.df):
                item = self.df.iloc[idx]
                results.append({
                    'Rank': i + 1,
                    'Distance': float(distances[0][i]),
                    'Section': item['Section'],
                    'Description': item['Description'],
                })
        return pd.DataFrame(results)

# --- Initialization ---
df = load_data(CSV_FILENAME)
engine = BNSSearchEngine(df, MODEL_NAME)
preprocessor = FIRPreprocessor(CRIME_KEYWORDS)

print("\nBlock 3 execution complete: FAISS Engine initialized and ready for search.")
```

```
File 'testing1.csv' not found.
Please upload the CSV file (e.g., 'testing1.csv') now:
Choose Files testing1.csv
• testing1.csv(text/csv) - 593175 bytes, last modified: 12/17/2025 - 100% done
Saving testing1.csv to testing1.csv
Successfully loaded 358 BNS sections.

[STEP 1/3] Loading Sentence-BERT model...
modules.json:  100%                                   349/349  [00:00<00:00,  28.5kB/s]
config_sentence_transformers.json:  100%              116/116  [00:00<00:00,  7.84kB/s]
README.md:         10.5k/?  [00:00<00:00,  908kB/s]
sentence_bert_config.json:  100%                      53.0/53.0  [00:00<00:00,  5.37kB/s]
config.json:  100%                                    612/612  [00:00<00:00,  27.5kB/s]
model.safetensors:  100%                         90.9M/90.9M  [00:05<00:00,  22.1MB/s]
tokenizer_config.json:  100%                          350/350  [00:00<00:00,  34.8kB/s]
vocab.txt:         232k/?  [00:00<00:00,  7.87MB/s]
tokenizer.json:         466k/?  [00:00<00:00,  25.2MB/s]
special_tokens_map.json:  100%                        112/112  [00:00<00:00,  10.3kB/s]
config.json:  100%                                    190/190  [00:00<00:00,  24.0kB/s]
[STEP 2/3] Generating embeddings for BNS descriptions...
[STEP 3/3] FAISS Index built with 358 vectors of dimension 384.

Block 3 execution complete: FAISS Engine initialized and ready for search.
```

```
# ---------------------------------------------------------
# Interactive UI Block (Replaces Blocks 4 & 5)
# Uses ipywidgets to create an expandable input field and a button
# for a better user experience in Google Colab.
# ---------------------------------------------------------

import ipywidgets as widgets
from IPython.display import display, clear_output
import pandas as pd

# Define the function that performs the search workflow (from previous logic)
def run_search_workflow(raw_fir, k=K_RESULTS):
    """
    Takes a raw FIR string, normalizes it, searches FAISS, and returns results DataFrame.
    Takes a raw FIR string, normalizes it, searches FAISS, and returns results DataFrame.
    """
    if not raw_fir.strip():
        return None, [], "Please enter a non-empty FIR report."

    # 1. Normalize
    normalized_query, detected_crimes = preprocessor.normalize(raw_fir)

    # 2. Search
    results_df = engine.search(normalized_query, k=k)

    summary = (
        f"-> Detected Crime Keywords: {detected_crimes}\n"
        f"-> Normalized Search Query: '{normalized_query}'"
    )
    return results_df, detected_crimes, summary

# ----------------------------------------
# WIDGET SETUP
# ----------------------------------------

# 1. Expandable Text Input (for FIR Report)
fir_input = widgets.Textarea(
    value='',
    placeholder='Paste the full, raw FIR report here (e.g., dates, times, narrative, etc.)',
    description='Raw FIR Input:',
    disabled=False,
    layout=widgets.Layout(width='100%', height='150px')
)

# 2. Search Button
search_button = widgets.Button(
    description='Predict BNS Sections',
    disabled=False,
    button_style='success', # Green color for action
    tooltip='Click to process the FIR and search the FAISS index.',
    icon='search'
)

# 3. Output Area
output_area = widgets.Output()

# ----------------------------------------
# EVENT HANDLER
# ----------------------------------------

def on_search_button_clicked(b):
    """Callback function when the button is clicked."""
    raw_fir = fir_input.value

    # Clear previous output
    with output_area:
        clear_output()
        print("Processing...")

        # Run the core logic
        results_df, detected_crimes, summary = run_search_workflow(raw_fir)

        # Display results or error
        if results_df is None:
            print(summary)
        else:
            print(summary)
            print("\n" + "="*50)
            print(f"TOP {K_RESULTS} PREDICTED BNS SECTIONS")
            print("="*50)

            # Use HTML display for nicer table formatting in Colab
            pd.set_option('display.max_colwidth', 100)

            # Select and rename columns for final display
            display_df = results_df[['Rank', 'Section', 'Distance', 'Description']]

            # Display the DataFrame
            display(display_df)

# Link the button to the click handler
search_button.on_click(on_search_button_clicked)

# ----------------------------------------
# EXECUTION (Display the UI)
# ----------------------------------------
```

```
print("\n" + "="*60)
print("BNS PREDICTION SYSTEM READY (Interactive Mode)")
print("="*60)

# Arrange widgets vertically
ui = widgets.VBox([
    fir_input,
    search_button,
    output_area
])

display(ui)
```

```
============================================================
BNS PREDICTION SYSTEM READY (Interactive Mode)
============================================================
```

Raw FIR In…

```
A large-scale corruption case emerged involving bribery and bribing within a government administration office.
Officials were accused of embezzlement, cheating, and criminal breach of trust. Counterfeit documents and forgery were
used to justify false claims and false statements. A data breach exposed money laundering networks linked to organized
crime. The accused attempted concealment and perjury, but digital audits confirmed fraud and identity theft.
```

Predict BNS Sectio…

```
'rocessing...
-> Detected Crime Keywords: ['theft', 'money', 'corruption', 'money laundering', 'counterfeit', 'concealment', 'data
oreach', 'organized crime', 'fraud', 'false statement', 'bribing', 'criminal breach of trust', 'forgery', 'perjury',
'bribery', 'false claim', 'identity theft', 'cheating', 'embezzlement']
-> Normalized Search Query: 'Crime Categories: perjury, theft, bribery, false claim, identity theft, cheating, false
statement, bribing, counterfeit, criminal breach of trust, money, concealment, data breach, organized crime, corruption,
embezzlement, fraud, forgery, money laundering. Context: A large-scale corruption case emerged involving bribery and
oribing within a government administration office. Officials were accused of embezzlement, cheating, and criminal breach of
:rust. Counterfeit documents and forgery were used to justify false claims and false statements. A data breach exposed
noney laundering networks linked to organized crime. The accused attempted concealment and perjury, but digital audits
confirmed fraud and identity theft.'

==================================================
TOP 5 PREDICTED BNS SECTIONS
==================================================
```

	Rank	Section	Distance	Description
0	1	BNS_233	0.743339	BNS_233 - According to BNS 233 Whoever corruptly uses or attempts to use as true or genuine evid...
1	2	BNS_318	0.878033	BNS_318 - According to BNS 318 (1) Whoever, by deceiving any person, fraudulently or dishonestly...
2	3	BNS_340	0.901238	BNS_340 - According to BNS 340 (1) A false document or electronic record made wholly or in part ...
3	4	BNS_112	0.911490	BNS_112 - According to BNS 112 (1) Whoever, being a member of a group or gang, either singly or ...
4	5	BNS_228	0.915944	BNS_228 - According to BNS 228 Whoever causes any circumstance to exist or makes any false entry...