CMPE 256 Recommender Systems Data Science Hackathon

❖ Clinical AI Assistant

❖ Overview :-

For the second part of the CMPE 256 Data Science Hackathon, my team and I built a Clinical AI Assistant—a Retrieval-Augmented Generation (RAG) system designed to answer clinical research questions using a large collection of medical research PDFs and clinical trial datasets. Our objective was to create an AI assistant that could provide grounded, data-driven responses strictly from the uploaded files without relying on external internet sources. We integrated multiple domain datasets covering COVID-19, Diabetes, Heart Attack, and Knee Injuries, along with corresponding CSV summaries of clinical studies. Using Google Colab with a T4 GPU, we implemented a pipeline that extracted text from thousands of PDF pages, chunked and embedded them with MiniLM Sentence Transformers, and stored them efficiently in a Chroma vector database on Google Drive for fast retrieval. We also developed a Gradio-based web interface that allows users to ask natural language questions and receive concise, evidence-grounded answers with exact PDF file names and page citations. This system demonstrated how RAG architectures can be applied in the healthcare domain to improve access to scientific knowledge and support clinicians and researchers in evidence-based decision-making.

❖ Technical Implementation :-

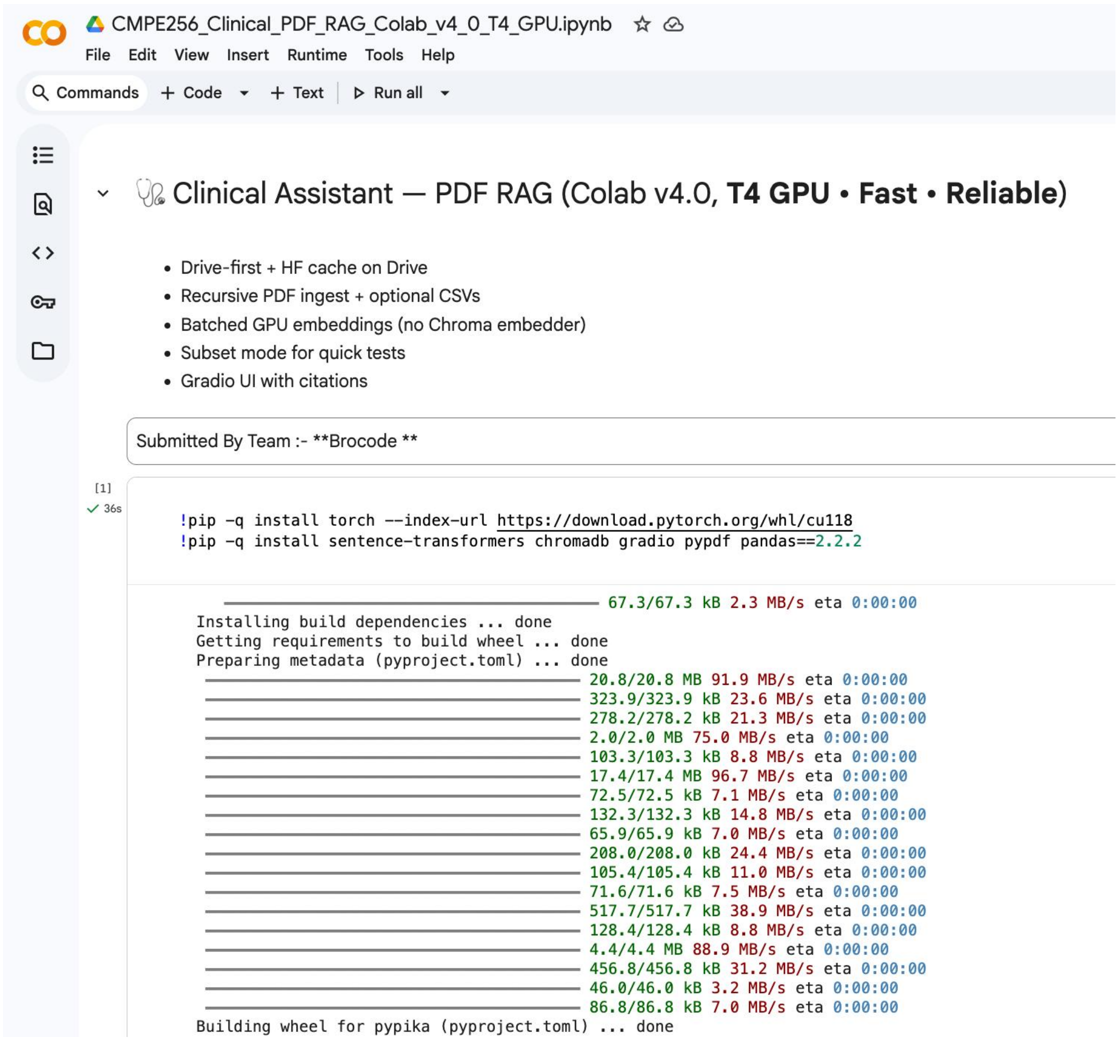
To develop the Clinical AI Assistant, my team and I designed a pipeline that could efficiently process, index, and retrieve relevant medical information from a large corpus of research PDFs and CSV datasets. We started by organizing all files into separate folders for each medical domain—COVID-19, Diabetes, Heart Attack, and Knee Injuries—and stored them in Google Drive for persistence. Using Google Colab with a T4 GPU runtime, we implemented the system entirely in Python.

The first major step involved data ingestion and text extraction. We used the pypdf library to recursively scan all subfolders, extract readable text from each page, and handle errors gracefully for scanned or corrupted PDFs. Next, the text was split into meaningful chunks (around 1,800 tokens each) to balance retrieval precision and embedding efficiency. Each chunk was then embedded using SentenceTransformer's MiniLM-L6-v2 model running on the GPU, which converted the text into dense numerical vectors suitable for semantic search.

We stored all embeddings, metadata (such as file name and page number), and raw text in a Chroma vector database, which we configured to persist on Google Drive so the assistant could retain knowledge across sessions. For real-time interaction, we built a Gradio web interface that allows users to enter natural-language clinical questions. The assistant retrieves the most semantically similar text chunks from the vector store using the query embeddings, then displays concise, grounded answers along with exact PDF file names and page citations.

To ensure scalability, we implemented batch embedding and incremental indexing, which significantly reduced processing time even for datasets containing tens of thousands of document chunks. Through this design, we successfully created an end-to-end RAG pipeline capable of running efficiently on Colab's GPU hardware while maintaining strict data grounding within the uploaded clinical documents.

❖ Screenshots :-



CMPE256_Clinical_PDF_RAG_Colab_v4_0_T4_GPU.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all

Clinical Assistant — PDF RAG (Colab v4.0, **T4 GPU • Fast • Reliable**)

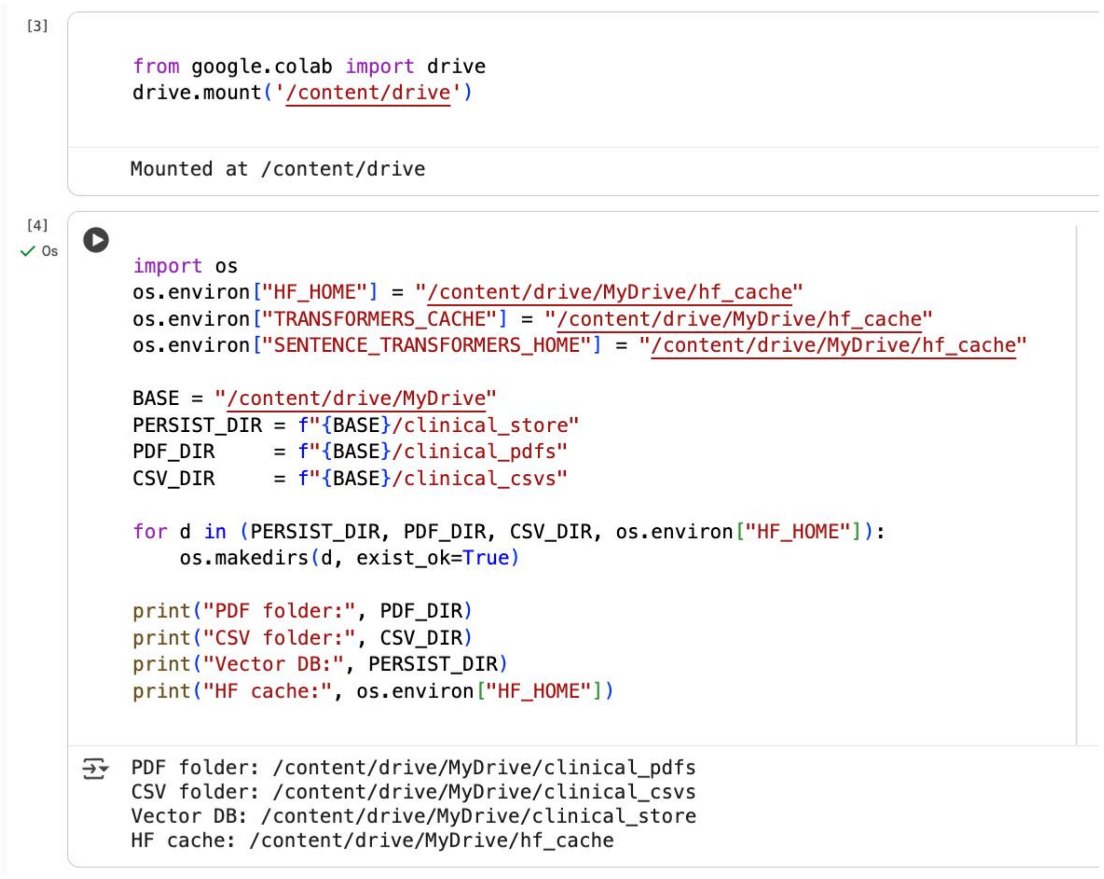
- Drive-first + HF cache on Drive
- Recursive PDF ingest + optional CSVs
- Batched GPU embeddings (no Chroma embedder)
- Subset mode for quick tests
- Gradio UI with citations

Submitted By Team :- **Brocode **

[1] 36s

```
!pip -q install torch --index-url https://download.pytorch.org/whl/cu118
!pip -q install sentence-transformers chromadb gradio pypdf pandas==2.2.2
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 67.3/67.3 kB 2.3 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 20.8/20.8 MB 91.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 323.9/323.9 kB 23.6 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 278.2/278.2 kB 21.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 75.0 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 103.3/103.3 kB 8.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.4/17.4 MB 96.7 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 72.5/72.5 kB 7.1 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 132.3/132.3 kB 14.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 65.9/65.9 kB 7.0 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 208.0/208.0 kB 24.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 105.4/105.4 kB 11.0 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 71.6/71.6 kB 7.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 517.7/517.7 kB 38.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 128.4/128.4 kB 8.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.4/4.4 MB 88.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 456.8/456.8 kB 31.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 46.0/46.0 kB 3.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 86.8/86.8 kB 7.0 MB/s eta 0:00:00
  Building wheel for pypika (pyproject.toml) ... done
```



[3]

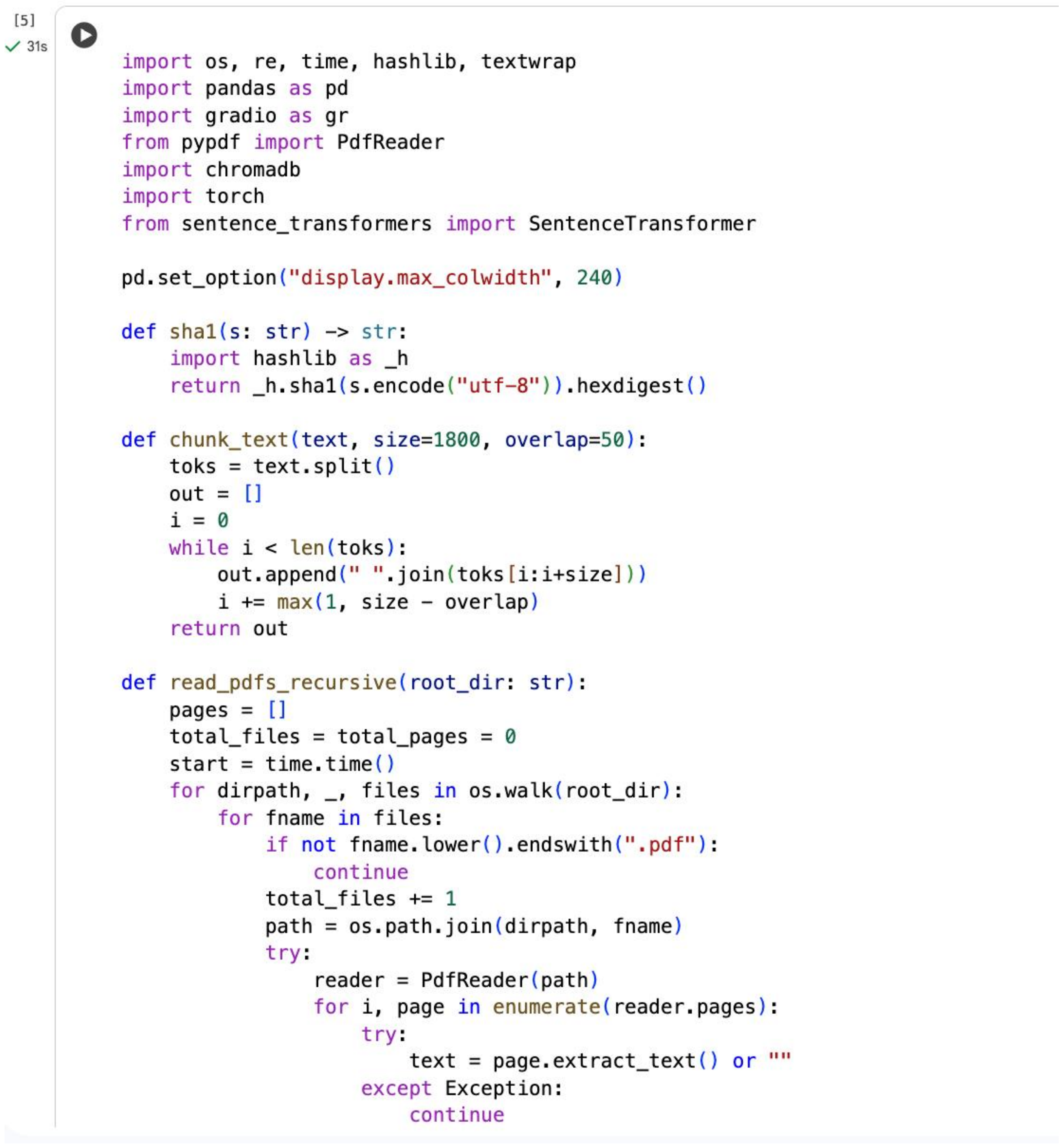
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[4] 0s

```
import os
os.environ["HF_HOME"] = "/content/drive/MyDrive/hf_cache"
os.environ["TRANSFORMERS_CACHE"] = "/content/drive/MyDrive/hf_cache"
os.environ["SENTENCE_TRANSFORMERS_HOME"] = "/content/drive/MyDrive/hf_cache"

BASE = "/content/drive/MyDrive"
PERSIST_DIR = f"{BASE}/clinical_store"
PDF_DIR     = f"{BASE}/clinical_pdfs"
CSV_DIR     = f"{BASE}/clinical_csvs"

for d in (PERSIST_DIR, PDF_DIR, CSV_DIR, os.environ["HF_HOME"]):
    os.makedirs(d, exist_ok=True)

print("PDF folder:", PDF_DIR)
print("CSV folder:", CSV_DIR)
print("Vector DB:", PERSIST_DIR)
print("HF cache:", os.environ["HF_HOME"])
```

```
PDF folder: /content/drive/MyDrive/clinical_pdfs
CSV folder: /content/drive/MyDrive/clinical_csvs
Vector DB: /content/drive/MyDrive/clinical_store
HF cache: /content/drive/MyDrive/hf_cache
```



```python
import os, re, time, hashlib, textwrap
import pandas as pd
import gradio as gr
from pypdf import PdfReader
import chromadb
import torch
from sentence_transformers import SentenceTransformer

pd.set_option("display.max_colwidth", 240)

def sha1(s: str) -> str:
    import hashlib as _h
    return _h.sha1(s.encode("utf-8")).hexdigest()

def chunk_text(text, size=1800, overlap=50):
    toks = text.split()
    out = []
    i = 0
    while i < len(toks):
        out.append(" ".join(toks[i:i+size]))
        i += max(1, size - overlap)
    return out

def read_pdfs_recursive(root_dir: str):
    pages = []
    total_files = total_pages = 0
    start = time.time()
    for dirpath, _, files in os.walk(root_dir):
        for fname in files:
            if not fname.lower().endswith(".pdf"):
                continue
            total_files += 1
            path = os.path.join(dirpath, fname)
            try:
                reader = PdfReader(path)
                for i, page in enumerate(reader.pages):
                    try:
                        text = page.extract_text() or ""
                    except Exception:
                        continue
```

[5]
✓ 31s

```
                        continue
                    text = re.sub(r"\s+", " ", text).strip()
                    if text:
                        pages.append({"source": fname, "page": i+1, "text": text})
                        total_pages += 1
                        if total_pages % 200 == 0:
                            print(f"Read pages: {total_pages} (files: {total_files})")
            except Exception as e:
                print("PDF read error:", fname, e)
    print(f"Finished PDFs — files: {total_files}, pages: {total_pages}, time: {time.time()-start:.1f}s")
    return pages

def read_csvs(csv_dir: str):
    csvs = [p for p in os.listdir(csv_dir) if p.lower().endswith('.csv')]
    rows = []
    for fname in csvs:
        path = os.path.join(csv_dir, fname)
        try:
            df = pd.read_csv(path)
            for i, row in df.iterrows():
                parts = [f"{col}: {str(val).strip()}" for col, val in row.items() if pd.notna(val)]
                text = " | ".join(parts)
                rows.append({"source": fname, "row": int(i), "text": text})
        except Exception as e:
            print("CSV read error:", fname, e)
    return rows

USE_SUBSET = False
TARGET_FOLDERS = {"Covid", "Heart_attack"}

def read_pdfs_subset(root_dir: str, targets):
    pages = []
    for dirpath, _, files in os.walk(root_dir):
        parts = set(dirpath.split(os.sep))
        if not (parts & targets):
            continue
        for fname in files:
            if not fname.lower().endswith(".pdf"):
                continue
            path = os.path.join(dirpath, fname)
            try:
                reader = PdfReader(path)
                for i, page in enumerate(reader.pages):
                    try:
                        text = page.extract_text() or ""
                    except Exception:
```

[5]
✓ 31s

```
                    try:
                        text = page.extract_text() or ""
                    except Exception:
```

```
                        continue
                    text = re.sub(r"\s+", " ", text).strip()
                    if text:
                        pages.append({"source": fname, "page": i+1, "text": text})
            except Exception as e:
                print("PDF read error:", fname, e)
    return pages
```

```
/usr/local/lib/python3.12/dist-packages/transformers/utils/hub.py:110: FutureWarning: Using `TRANSFORMERS_CACHE` is deprecated and will be removed in v5 of Transformers. Use `HF_HOME` instead.
  warnings.warn(
```

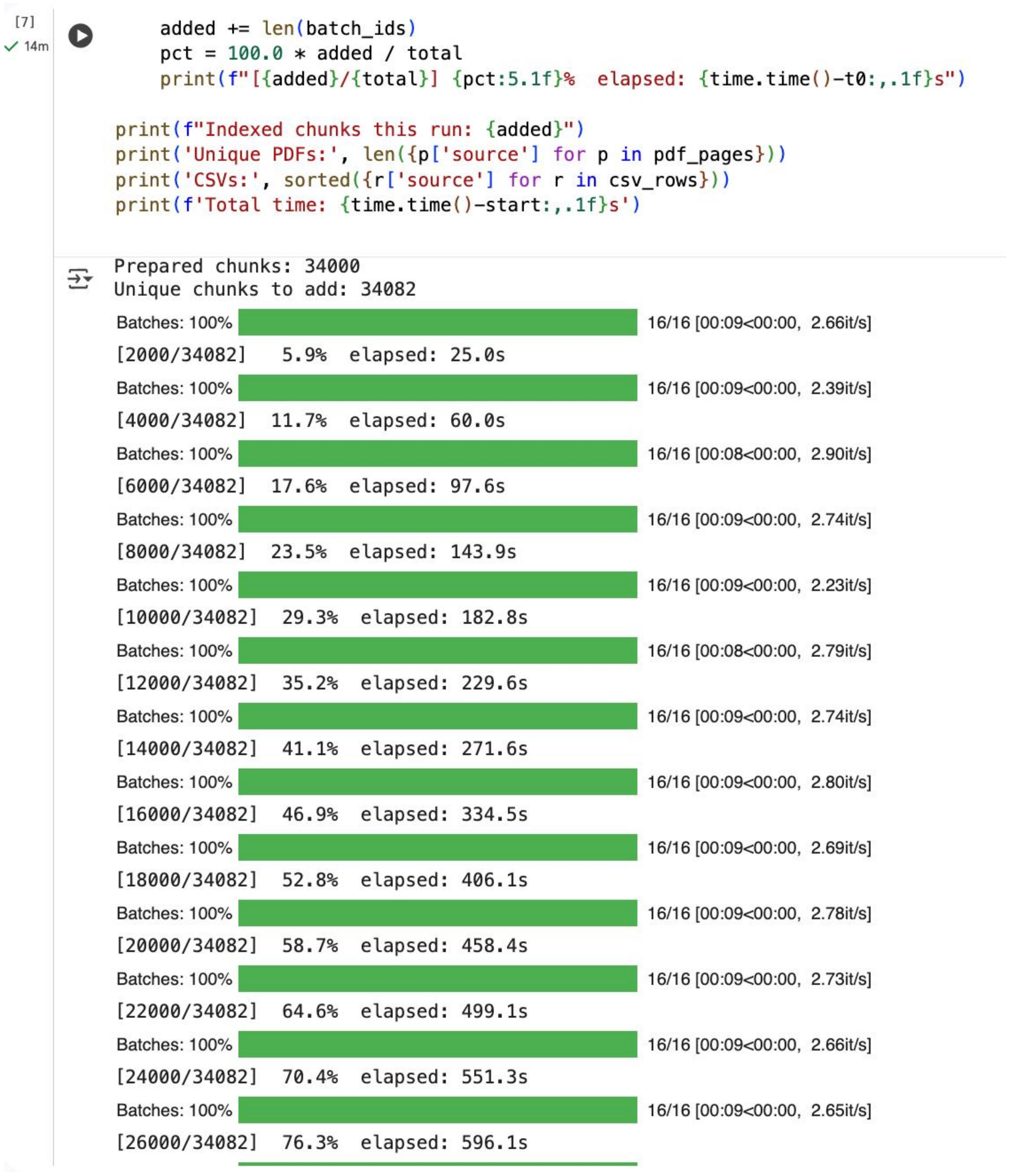
[6]
✓ 17s

```python
device = "cuda" if torch.cuda.is_available() else "cpu"
print("Embedding device:", device)
st_model = SentenceTransformer("sentence-transformers/all-MiniLM-L6-v2", device=device)
ENC_BATCH = 128

client = chromadb.PersistentClient(path=PERSIST_DIR)
# Clean rebuild toggle
CLEAN_REBUILD = False
if CLEAN_REBUILD:
    try:
        client.delete_collection("clinical_pdf_store")
        print("Deleted old collection")
    except Exception:
        pass
try:
    collection = client.get_collection("clinical_pdf_store")
    print("Collection opened:", collection.name)
except Exception:
    collection = client.create_collection(name="clinical_pdf_store")
    print("Collection created:", collection.name)
```

```
Embedding device: cuda
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public model
  warnings.warn(
Collection opened: clinical_pdf_store
```

[7]
✓ 14m

```python
start = time.time()

pdf_pages = read_pdfs_subset(PDF_DIR, TARGET_FOLDERS) if USE_SUBSET else read_pdfs_recursive(PDF_DIR)
csv_rows = read_csvs(CSV_DIR)

ids, docs, metas = [], [], []
chunk_count = 0

def push_chunks(uid_prefix, meta_obj, text):
    global chunk_count
    for ch in chunk_text(text):
        uid = f"{uid_prefix}::h{sha1(ch)[:12]}"
        ids.append(uid); docs.append(ch); metas.append(meta_obj)
        chunk_count += 1
        if chunk_count % 500 == 0:
            print(f"Prepared chunks: {chunk_count}")

for p in pdf_pages:
    push_chunks(f"pdf::{p['source']}::p{p['page']}", {"source": p["source"], "page": p["page"], "type":"pdf"}, p["text"])
for r in csv_rows:
    push_chunks(f"csv::{r['source']}::row{r['row']}", {"source": r["source"], "row": r["row"], "type":"csv"}, r["text"])

seen = set(); I, D, M = [], [], []
for i, d, m in zip(ids, docs, metas):
    if i in seen:
        continue
    seen.add(i); I.append(i); D.append(d); M.append(m)

print(f"Unique chunks to add: {len(I)}")

BATCH_ADD = 2000
added = 0; t0 = time.time(); total = len(I)

for s in range(0, total, BATCH_ADD):
    e = min(s + BATCH_ADD, total)
    batch_ids, batch_docs, batch_metas = I[s:e], D[s:e], M[s:e]

    batch_embs = st_model.encode(
        batch_docs, device=device, batch_size=ENC_BATCH,
        show_progress_bar=True, convert_to_numpy=True, normalize_embeddings=True
    ).tolist()

    collection.add(ids=batch_ids, embeddings=batch_embs, metadatas=batch_metas, documents=batch_docs)
```


[7]
✓ 14m

```
        added += len(batch_ids)
        pct = 100.0 * added / total
        print(f"[{added}/{total}] {pct:5.1f}%  elapsed: {time.time()-t0:,.1f}s")

print(f"Indexed chunks this run: {added}")
print('Unique PDFs:', len({p['source'] for p in pdf_pages}))
print('CSVs:', sorted({r['source'] for r in csv_rows}))
print(f'Total time: {time.time()-start:,.1f}s')
```

```
Prepared chunks: 34000
Unique chunks to add: 34082
Batches: 100%  16/16 [00:09<00:00,  2.66it/s]
[2000/34082]   5.9%  elapsed: 25.0s
Batches: 100%  16/16 [00:09<00:00,  2.39it/s]
[4000/34082]  11.7%  elapsed: 60.0s
Batches: 100%  16/16 [00:08<00:00,  2.90it/s]
[6000/34082]  17.6%  elapsed: 97.6s
Batches: 100%  16/16 [00:09<00:00,  2.74it/s]
[8000/34082]  23.5%  elapsed: 143.9s
Batches: 100%  16/16 [00:09<00:00,  2.23it/s]
[10000/34082]  29.3%  elapsed: 182.8s
Batches: 100%  16/16 [00:08<00:00,  2.79it/s]
[12000/34082]  35.2%  elapsed: 229.6s
Batches: 100%  16/16 [00:09<00:00,  2.74it/s]
[14000/34082]  41.1%  elapsed: 271.6s
Batches: 100%  16/16 [00:09<00:00,  2.80it/s]
[16000/34082]  46.9%  elapsed: 334.5s
Batches: 100%  16/16 [00:09<00:00,  2.69it/s]
[18000/34082]  52.8%  elapsed: 406.1s
Batches: 100%  16/16 [00:09<00:00,  2.78it/s]
[20000/34082]  58.7%  elapsed: 458.4s
Batches: 100%  16/16 [00:09<00:00,  2.73it/s]
[22000/34082]  64.6%  elapsed: 499.1s
Batches: 100%  16/16 [00:09<00:00,  2.66it/s]
[24000/34082]  70.4%  elapsed: 551.3s
Batches: 100%  16/16 [00:09<00:00,  2.65it/s]
[26000/34082]  76.3%  elapsed: 596.1s
```

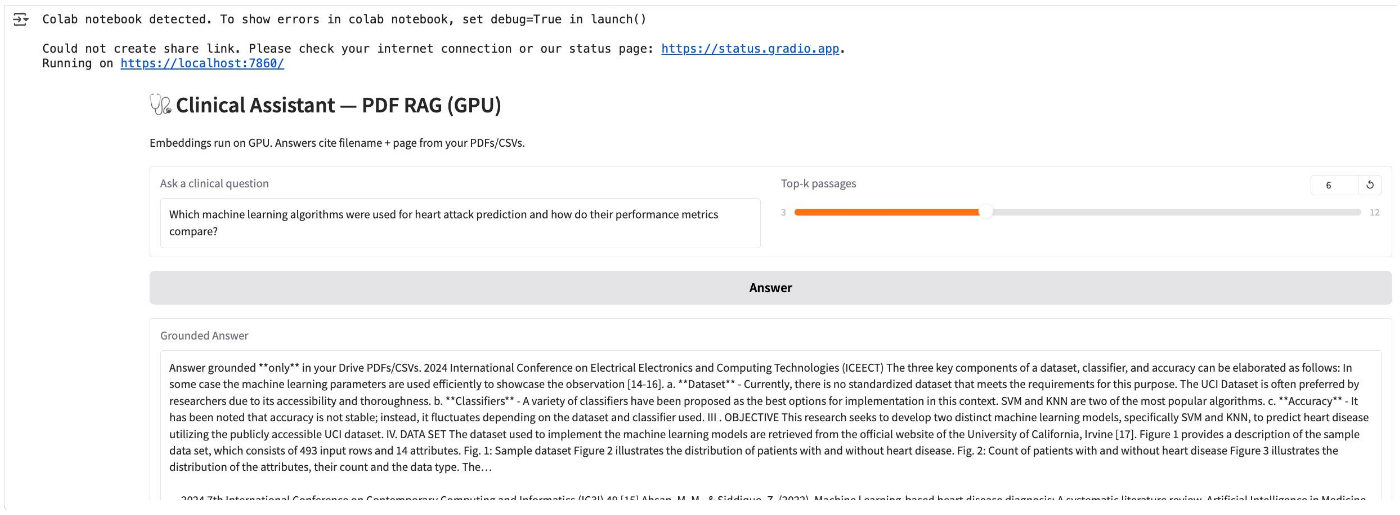
```
[18000/34082]  52.8%  elapsed: 406.1s
Batches: 100% ████████████████████ 16/16 [00:09<00:00,  2.78it/s]
[20000/34082]  58.7%  elapsed: 458.4s
Batches: 100% ████████████████████ 16/16 [00:09<00:00,  2.73it/s]
[22000/34082]  64.6%  elapsed: 499.1s
Batches: 100% ████████████████████ 16/16 [00:09<00:00,  2.66it/s]
[24000/34082]  70.4%  elapsed: 551.3s
Batches: 100% ████████████████████ 16/16 [00:09<00:00,  2.65it/s]
[26000/34082]  76.3%  elapsed: 596.1s
Batches: 100% ████████████████████ 16/16 [00:08<00:00,  2.45it/s]
[28000/34082]  82.2%  elapsed: 650.5s
Batches: 100% ████████████████████ 16/16 [00:09<00:00,  2.25it/s]
[30000/34082]  88.0%  elapsed: 695.4s
Batches: 100% ████████████████████ 16/16 [00:09<00:00,  2.65it/s]
[32000/34082]  93.9%  elapsed: 788.9s
Batches: 100% ████████████████████ 16/16 [00:09<00:00,  2.65it/s]
[34000/34082]  99.8%  elapsed: 836.3s
Batches: 100% ████████████████████ 1/1 [00:00<00:00,  2.22it/s]
[34082/34082] 100.0%  elapsed: 839.5s
Indexed chunks this run: 34082
Unique PDFs: 20
CSVs: ['ctg-studies_Hearattack.csv', 'ctg-studies_KneeInjuries.csv', 'ctg-studies_covid.csv', 'ctg-studies_diabetes.csv']
Total time: 866.6s
```

[8]
✓ 0s

```python
def format_sources(metas, max_src=8):
    seen = set(); out = []
    for m in metas:
        key = (m.get("source"), m.get("page"), m.get("row"))
        if key in seen:
            continue
        seen.add(key)
        if m.get("type") == "pdf":
            out.append(f"{m['source']} p. {m.get('page')}")
        else:
            out.append(f"{m['source']} row {m.get('row')}")
        if len(out) >= max_src: break
    return out

def answer(query, k=6, max_chars=1500):
    if not query.strip():
        return "Please type a question.", []
    q_emb = st_model.encode([query], device=device, batch_size=1,
                            show_progress_bar=False, convert_to_numpy=True, normalize_embeddings=True).tolist()
    res = collection.query(query_embeddings=q_emb, n_results=int(k))
    docs  = res.get("documents", [[]])[0] or []
    metas = res.get("metadatas", [[]])[0] or []
    if not docs:
        return "(No relevant passages found in your local PDFs/CSVs.)", []
    pieces = [textwrap.shorten(d.replace("\n"," "), width=max_chars, placeholder="…") for d in docs[:3]]
    header = "Answer grounded **only** in your Drive PDFs/CSVs. "
    return header + "\n\n- ".join(pieces), format_sources(metas)
```

[9]
✓ 0s

```python
with gr.Blocks(title="Clinical Assistant — PDF RAG (GPU)") as demo:
    gr.Markdown("# 🩺 Clinical Assistant — **PDF RAG (GPU)**")
    gr.Markdown("Embeddings run on GPU. Answers cite filename + page from your PDFs/CSVs.")
    with gr.Row():
        q = gr.Textbox(label="Ask a clinical question", placeholder="e.g., Which models achieved highest F1 for COVID detection?")
        k = gr.Slider(3, 12, value=6, step=1, label="Top-k passages")
    ask = gr.Button("Answer")
    ans = gr.Textbox(label="Grounded Answer", lines=12)
    src = gr.JSON(label="Sources (file & page)")

    def _go(query, topk):
        a, s = answer(query, k=topk)
        return a, s

    ask.click(_go, inputs=[q, k], outputs=[ans, src])

demo.launch(share=True)
```



```
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()

Could not create share link. Please check your internet connection or our status page: https://status.gradio.app.
Running on https://localhost:7860/
```

Clinical Assistant — PDF RAG (GPU)

Embeddings run on GPU. Answers cite filename + page from your PDFs/CSVs.

Ask a clinical question

Which machine learning algorithms were used for heart attack prediction and how do their performance metrics compare?

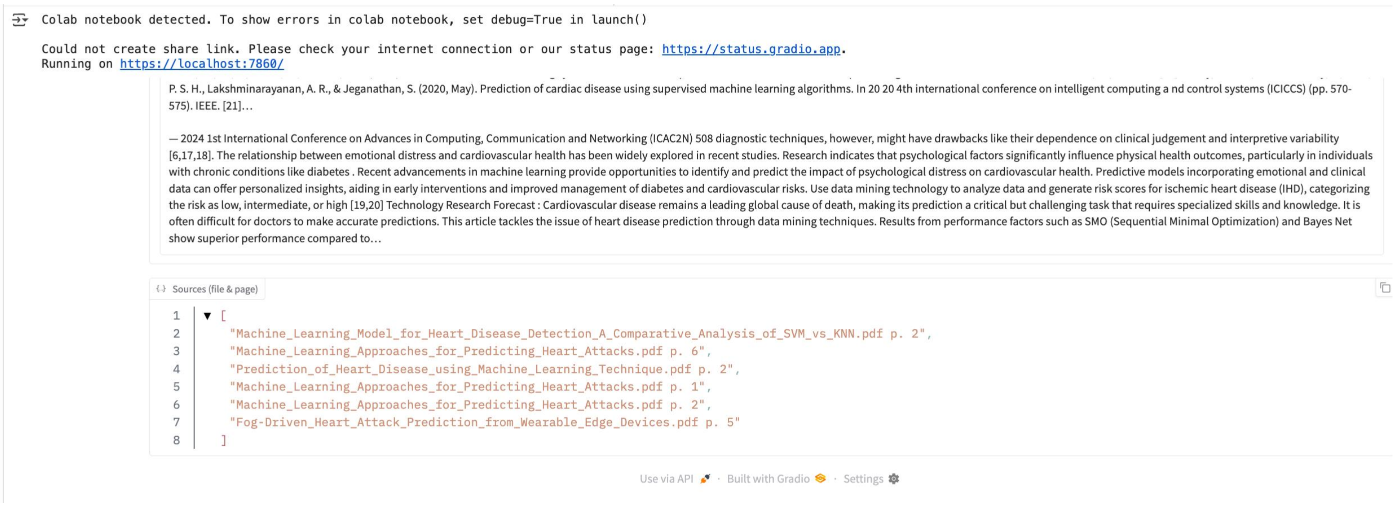
Top-k passages 6

Answer

Grounded Answer

Answer grounded **only** in your Drive PDFs/CSVs. 2024 International Conference on Electrical Electronics and Computing Technologies (ICEECT) The three key components of a dataset, classifier, and accuracy can be elaborated as follows: In some case the machine learning parameters are used efficiently to showcase the observation [14-16]. a. **Dataset** - Currently, there is no standardized dataset that meets the requirements for this purpose. The UCI Dataset is often preferred by researchers due to its accessibility and thoroughness. b. **Classifiers** - A variety of classifiers have been proposed as the best options for implementation in this context. SVM and KNN are two of the most popular algorithms. c. **Accuracy** - It has been noted that accuracy is not stable; instead, it fluctuates depending on the dataset and classifier used. III . OBJECTIVE This research seeks to develop two distinct machine learning models, specifically SVM and KNN, to predict heart disease utilizing the publicly accessible UCI dataset. IV. DATA SET The dataset used to implement the machine learning models are retrieved from the official website of the University of California, Irvine [17]. Figure 1 provides a description of the sample data set, which consists of 493 input rows and 14 attributes. Fig. 1: Sample dataset Figure 2 illustrates the distribution of patients with and without heart disease. Fig. 2: Count of patients with and without heart disease Figure 3 illustrates the distribution of the attributes, their count and the data type. The…

— 2024 7th International Conference on Contemporary Computing and Informatics (IC3I) 49 [15] Ahsan, M. M., & Siddique, Z. (2022). Machine Learning-based heart disease diagnosis: A systematic literature review. Artificial Intelligence in Medicine



```
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()

Could not create share link. Please check your internet connection or our status page: https://status.gradio.app.
Running on https://localhost:7860/
```

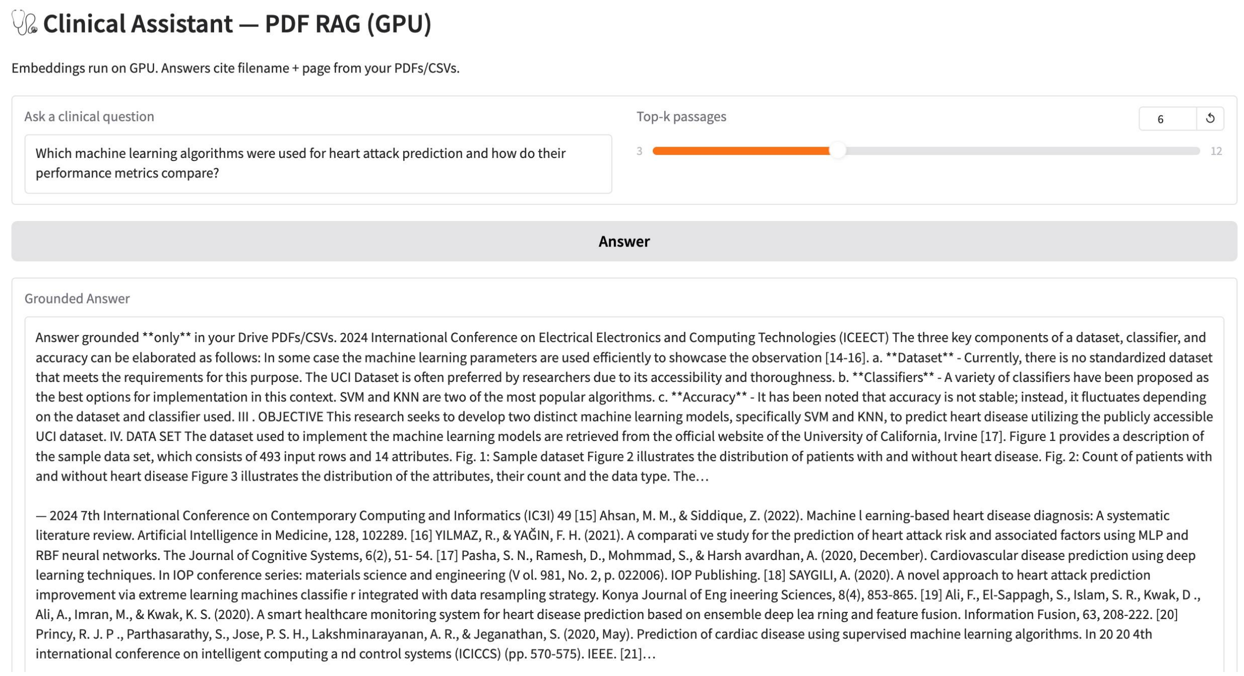
P. S. H., Lakshminarayanan, A. R., & Jeganathan, S. (2020, May). Prediction of cardiac disease using supervised machine learning algorithms. In 20 20 4th international conference on intelligent computing a nd control systems (ICICCS) (pp. 570-575). IEEE. [21]…

— 2024 1st International Conference on Advances in Computing, Communication and Networking (ICAC2N) 508 diagnostic techniques, however, might have drawbacks like their dependence on clinical judgement and interpretive variability [6,17,18]. The relationship between emotional distress and cardiovascular health has been widely explored in recent studies. Research indicates that psychological factors significantly influence physical health outcomes, particularly in individuals with chronic conditions like diabetes . Recent advancements in machine learning provide opportunities to identify and predict the impact of psychological distress on cardiovascular health. Predictive models incorporating emotional and clinical data can offer personalized insights, aiding in early interventions and improved management of diabetes and cardiovascular risks. Use data mining technology to analyze data and generate risk scores for ischemic heart disease (IHD), categorizing the risk as low, intermediate, or high [19,20] Technology Research Forecast : Cardiovascular disease remains a leading global cause of death, making its prediction a critical but challenging task that requires specialized skills and knowledge. It is often difficult for doctors to make accurate predictions. This article tackles the issue of heart disease prediction through data mining techniques. Results from performance factors such as SMO (Sequential Minimal Optimization) and Bayes Net show superior performance compared to…

Sources (file & page)

```
[
  "Machine_Learning_Model_for_Heart_Disease_Detection_A_Comparative_Analysis_of_SVM_vs_KNN.pdf p. 2",
  "Machine_Learning_Approaches_for_Predicting_Heart_Attacks.pdf p. 6",
  "Prediction_of_Heart_Disease_using_Machine_Learning_Technique.pdf p. 2",
  "Machine_Learning_Approaches_for_Predicting_Heart_Attacks.pdf p. 1",
  "Machine_Learning_Approaches_for_Predicting_Heart_Attacks.pdf p. 2",
  "Fog-Driven_Heart_Attack_Prediction_from_Wearable_Edge_Devices.pdf p. 5"
]
```

Use via API · Built with Gradio · Settings



Clinical Assistant — PDF RAG (GPU)

Embeddings run on GPU. Answers cite filename + page from your PDFs/CSVs.

Ask a clinical question

Which machine learning algorithms were used for heart attack prediction and how do their performance metrics compare?

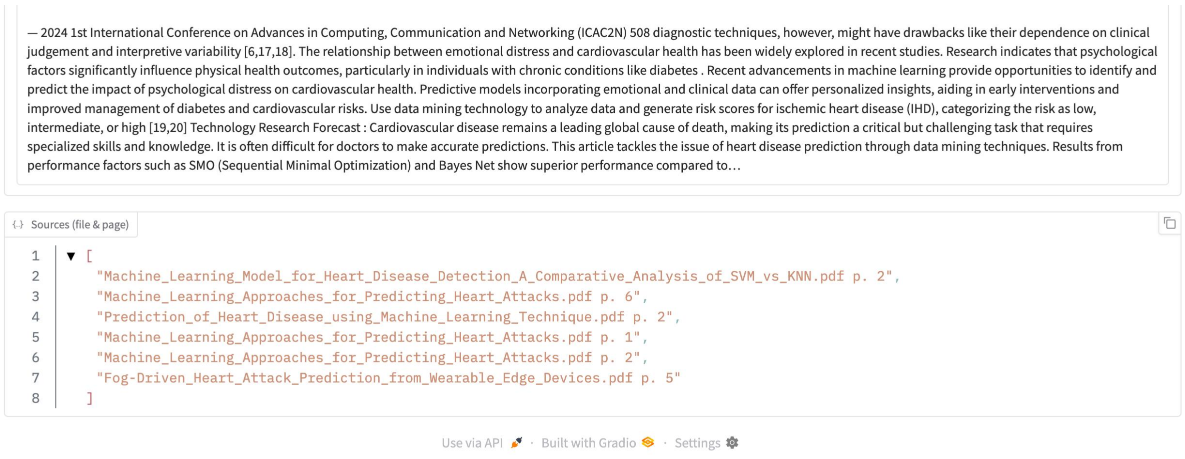
Top-k passages 6

Answer

Grounded Answer

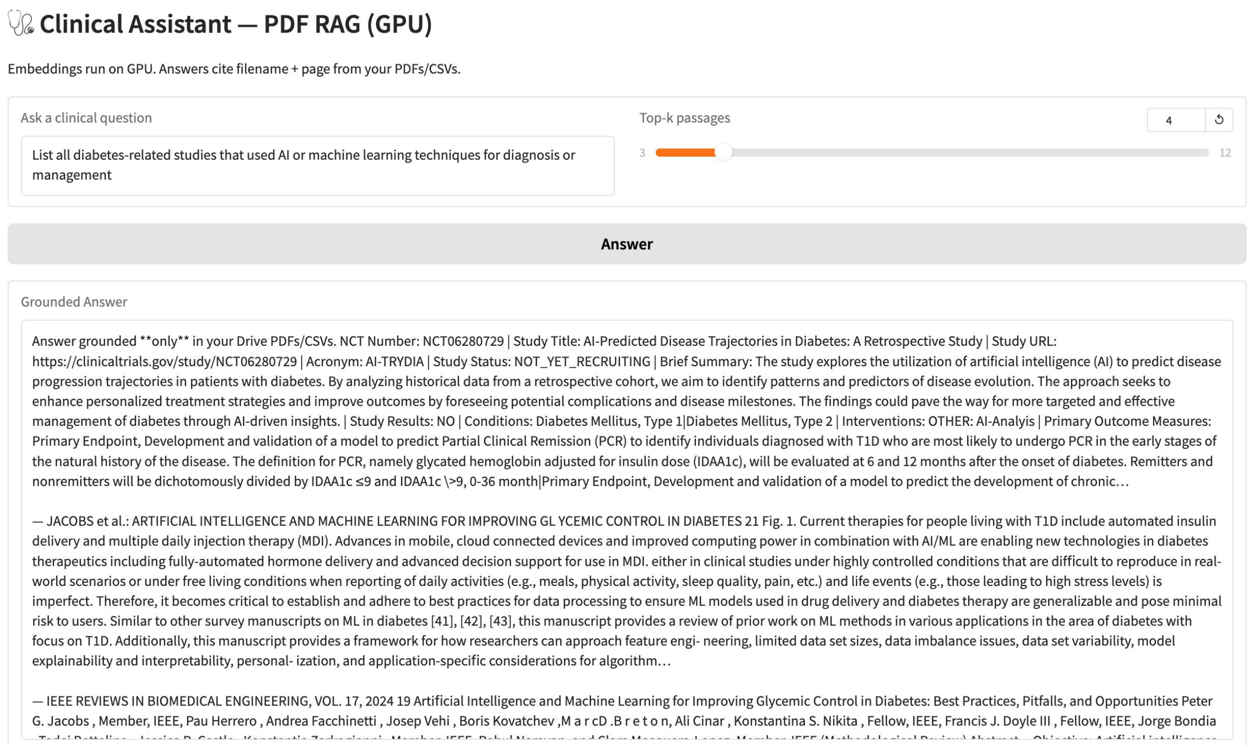
Answer grounded **only** in your Drive PDFs/CSVs. 2024 International Conference on Electrical Electronics and Computing Technologies (ICEECT) The three key components of a dataset, classifier, and accuracy can be elaborated as follows: In some case the machine learning parameters are used efficiently to showcase the observation [14-16]. a. **Dataset** - Currently, there is no standardized dataset that meets the requirements for this purpose. The UCI Dataset is often preferred by researchers due to its accessibility and thoroughness. b. **Classifiers** - A variety of classifiers have been proposed as the best options for implementation in this context. SVM and KNN are two of the most popular algorithms. c. **Accuracy** - It has been noted that accuracy is not stable; instead, it fluctuates depending on the dataset and classifier used. III . OBJECTIVE This research seeks to develop two distinct machine learning models, specifically SVM and KNN, to predict heart disease utilizing the publicly accessible UCI dataset. IV. DATA SET The dataset used to implement the machine learning models are retrieved from the official website of the University of California, Irvine [17]. Figure 1 provides a description of the sample data set, which consists of 493 input rows and 14 attributes. Fig. 1: Sample dataset Figure 2 illustrates the distribution of patients with and without heart disease. Fig. 2: Count of patients with and without heart disease Figure 3 illustrates the distribution of the attributes, their count and the data type. The…

— 2024 7th International Conference on Contemporary Computing and Informatics (IC3I) 49 [15] Ahsan, M. M., & Siddique, Z. (2022). Machine l earning-based heart disease diagnosis: A systematic literature review. Artificial Intelligence in Medicine, 128, 102289. [16] YILMAZ, R., & YAĞIN, F. H. (2021). A comparati ve study for the prediction of heart attack risk and associated factors using MLP and RBF neural networks. The Journal of Cognitive Systems, 6(2), 51- 54. [17] Pasha, S. N., Ramesh, D., Mohmmad, S., & Harsh avardhan, A. (2020, December). Cardiovascular disease prediction using deep learning techniques. In IOP conference series: materials science and engineering (V ol. 981, No. 2, p. 022006). IOP Publishing. [18] SAYGILI, A. (2020). A novel approach to heart attack prediction improvement via extreme learning machines classifie r integrated with data resampling strategy. Konya Journal of Eng ineering Sciences, 8(4), 853-865. [19] Ali, F., El-Sappagh, S., Islam, S. R., Kwak, D ., Ali, A., Imran, M., & Kwak, K. S. (2020). A smart healthcare monitoring system for heart disease prediction based on ensemble deep lea rning and feature fusion. Information Fusion, 63, 208-222. [20] Princy, R. J. P ., Parthasarathy, S., Jose, P. S. H., Lakshminarayanan, A. R., & Jeganathan, S. (2020, May). Prediction of cardiac disease using supervised machine learning algorithms. In 20 20 4th international conference on intelligent computing a nd control systems (ICICCS) (pp. 570-575). IEEE. [21]…



— 2024 1st International Conference on Advances in Computing, Communication and Networking (ICAC2N) 508 diagnostic techniques, however, might have drawbacks like their dependence on clinical judgement and interpretive variability [6,17,18]. The relationship between emotional distress and cardiovascular health has been widely explored in recent studies. Research indicates that psychological factors significantly influence physical health outcomes, particularly in individuals with chronic conditions like diabetes . Recent advancements in machine learning provide opportunities to identify and predict the impact of psychological distress on cardiovascular health. Predictive models incorporating emotional and clinical data can offer personalized insights, aiding in early interventions and improved management of diabetes and cardiovascular risks. Use data mining technology to analyze data and generate risk scores for ischemic heart disease (IHD), categorizing the risk as low, intermediate, or high [19,20] Technology Research Forecast : Cardiovascular disease remains a leading global cause of death, making its prediction a critical but challenging task that requires specialized skills and knowledge. It is often difficult for doctors to make accurate predictions. This article tackles the issue of heart disease prediction through data mining techniques. Results from performance factors such as SMO (Sequential Minimal Optimization) and Bayes Net show superior performance compared to…

```
▼ [
  "Machine_Learning_Model_for_Heart_Disease_Detection_A_Comparative_Analysis_of_SVM_vs_KNN.pdf p. 2",
  "Machine_Learning_Approaches_for_Predicting_Heart_Attacks.pdf p. 6",
  "Prediction_of_Heart_Disease_using_Machine_Learning_Technique.pdf p. 2",
  "Machine_Learning_Approaches_for_Predicting_Heart_Attacks.pdf p. 1",
  "Machine_Learning_Approaches_for_Predicting_Heart_Attacks.pdf p. 2",
  "Fog-Driven_Heart_Attack_Prediction_from_Wearable_Edge_Devices.pdf p. 5"
]
```



🩺 Clinical Assistant — PDF RAG (GPU)

Embeddings runs on GPU. Answers cite filename + page from your PDFs/CSVs.

Ask a clinical question: List all diabetes-related studies that used AI or machine learning techniques for diagnosis or management

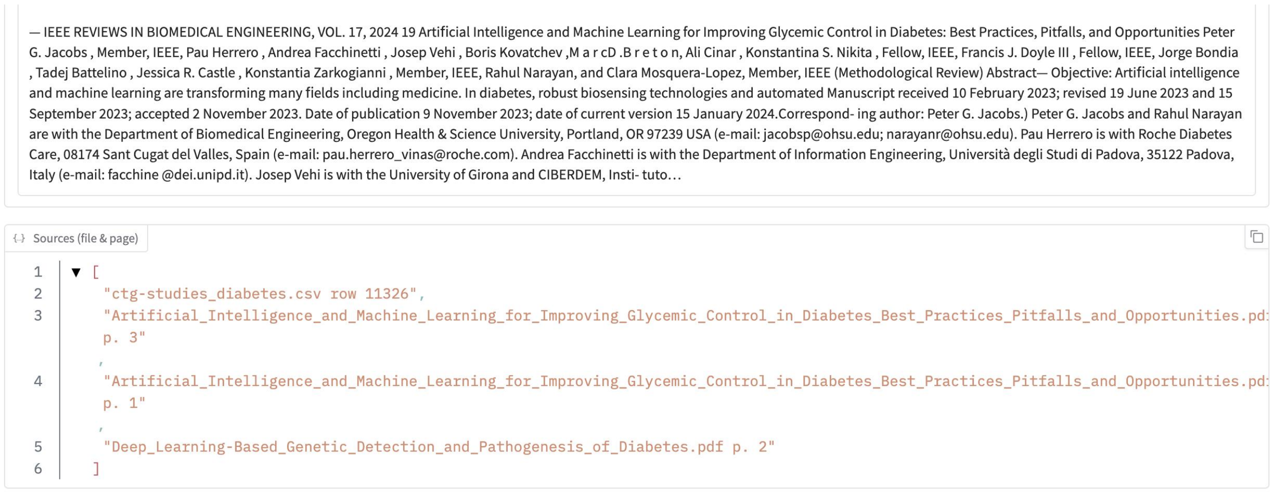
Top-k passages: 4 (3–12)

Grounded Answer

Answer grounded **only** in your Drive PDFs/CSVs. NCT Number: NCT06280729 | Study Title: AI-Predicted Disease Trajectories in Diabetes: A Retrospective Study | Study URL: https://clinicaltrials.gov/study/NCT06280729 | Acronym: AI-TRYDIA | Study Status: NOT_YET_RECRUITING | Brief Summary: The study explores the utilization of artificial intelligence (AI) to predict disease progression trajectories in patients with diabetes. By analyzing historical data from a retrospective cohort, we aim to identify patterns and predictors of disease evolution. The approach seeks to enhance personalized treatment strategies and improve outcomes by foreseeing potential complications and disease milestones. The findings could pave the way for more targeted and effective management of diabetes through AI-driven insights. | Study Results: NO | Conditions: Diabetes Mellitus, Type 1|Diabetes Mellitus, Type 2 | Interventions: OTHER: AI-Analyis | Primary Outcome Measures: Primary Endpoint, Development and validation of a model to predict Partial Clinical Remission (PCR) to identify individuals diagnosed with T1D who are most likely to undergo PCR in the early stages of the natural history of the disease. The definition for PCR, namely glycated hemoglobin adjusted for insulin dose (IDAA1c), will be evaluated at 6 and 12 months after the onset of diabetes. Remitters and nonremitters will be dichotomously divided by IDAA1c ≤9 and IDAA1c \>9, 0-36 month|Primary Endpoint, Development and validation of a model to predict the development of chronic…

— JACOBS et al.: ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING FOR IMPROVING GL YCEMIC CONTROL IN DIABETES 21 Fig. 1. Current therapies for people living with T1D include automated insulin delivery and multiple daily injection therapy (MDI). Advances in mobile, cloud connected devices and improved computing power in combination with AI/ML are enabling new technologies in diabetes therapeutics including fully-automated hormone delivery and advanced decision support for use in MDI. either in clinical studies under highly controlled conditions that are difficult to reproduce in real-world scenarios or under free living conditions when reporting of daily activities (e.g., meals, physical activity, sleep quality, pain, etc.) and life events (e.g., those leading to high stress levels) is imperfect. Therefore, it becomes critical to establish and adhere to best practices for data processing to ensure ML models used in drug delivery and diabetes therapy are generalizable and pose minimal risk to users. Similar to other survey manuscripts on ML in diabetes [41], [42], [43], this manuscript provides a review of prior work on ML methods in various applications in the area of diabetes with focus on T1D. Additionally, this manuscript provides a framework for how researchers can approach feature engi- neering, limited data set sizes, data imbalance issues, data set variability, model explainability and interpretability, personal- ization, and application-specific considerations for algorithm…

— IEEE REVIEWS IN BIOMEDICAL ENGINEERING, VOL. 17, 2024 19 Artificial Intelligence and Machine Learning for Improving Glycemic Control in Diabetes: Best Practices, Pitfalls, and Opportunities Peter G. Jacobs , Member, IEEE, Pau Herrero , Andrea Facchinetti , Josep Vehi , Boris Kovatchev ,M a r cD .B r e t o n, Ali Cinar , Konstantina S. Nikita , Fellow, IEEE, Francis J. Doyle III , Fellow, IEEE, Jorge Bondia



— IEEE REVIEWS IN BIOMEDICAL ENGINEERING, VOL. 17, 2024 19 Artificial Intelligence and Machine Learning for Improving Glycemic Control in Diabetes: Best Practices, Pitfalls, and Opportunities Peter G. Jacobs , Member, IEEE, Pau Herrero , Andrea Facchinetti , Josep Vehi , Boris Kovatchev ,M a r cD .B r e t o n, Ali Cinar , Konstantina S. Nikita , Fellow, IEEE, Francis J. Doyle III , Fellow, IEEE, Jorge Bondia , Tadej Battelino , Jessica R. Castle , Konstantia Zarkogianni , Member, IEEE, Rahul Narayan, and Clara Mosquera-Lopez, Member, IEEE (Methodological Review) Abstract— Objective: Artificial intelligence and machine learning are transforming many fields including medicine. In diabetes, robust biosensing technologies and automated Manuscript received 10 February 2023; revised 19 June 2023 and 15 September 2023; accepted 2 November 2023. Date of publication 9 November 2023; date of current version 15 January 2024.Correspond- ing author: Peter G. Jacobs.) Peter G. Jacobs and Rahul Narayan are with the Department of Biomedical Engineering, Oregon Health & Science University, Portland, OR 97239 USA (e-mail: jacobsp@ohsu.edu; narayanr@ohsu.edu). Pau Herrero is with Roche Diabetes Care, 08174 Sant Cugat del Valles, Spain (e-mail: pau.herrero_vinas@roche.com). Andrea Facchinetti is with the Department of Information Engineering, Università degli Studi di Padova, 35122 Padova, Italy (e-mail: facchine @dei.unipd.it). Josep Vehi is with the University of Girona and CIBERDEM, Insti- tuto…

```
▼ [
  "ctg-studies_diabetes.csv row 11326",
  "Artificial_Intelligence_and_Machine_Learning_for_Improving_Glycemic_Control_in_Diabetes_Best_Practices_Pitfalls_and_Opportunities.pd
  p. 3"
  ,
  "Artificial_Intelligence_and_Machine_Learning_for_Improving_Glycemic_Control_in_Diabetes_Best_Practices_Pitfalls_and_Opportunities.pd
  p. 1"
  ,
  "Deep_Learning-Based_Genetic_Detection_and_Pathogenesis_of_Diabetes.pdf p. 2"
]
```