```
import re
import numpy as np
import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.metrics import silhouette_score


try:
    from sentence_transformers import SentenceTransformer
except ImportError:
    !pip install -q sentence-transformers
    from sentence_transformers import SentenceTransformer
```

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

import os

EXCEL_PATH = "/content/drive/MyDrive/Harvard HW#5/derm.xlsx"

print("Using path:", EXCEL_PATH)
print("File exists:", os.path.exists(EXCEL_PATH))
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Using path: /content/drive/MyDrive/Harvard HW#5/derm.xlsx
File exists: True
```

```
Start coding or generate with AI.
```

```
# Load and expload inclusion criteria

def load_and_expand_criteria(
    path: str,
    id_col: str = "nctid",
    crit_col: str = "inclusion"
) -> pd.DataFrame:

    df_raw = pd.read_excel(path)

    print("Columns:", df_raw.columns.tolist())

    df = df_raw[[id_col, crit_col]].copy()

    df[crit_col] = df[crit_col].astype(str)

    rows = []
    splitter = re.compile(r"(\n|\r|\r\n|\*)")

    for _, row in df.iterrows():
        nctid = row[id_col]
        text = row[crit_col]

        raw_parts = [p.strip() for p in splitter.split(text)
                     if p.strip() and p.strip() != "*"]

        cleaned_parts = []
        for p in raw_parts:
            p = re.sub(r"^[\-\*\u2022]?\s*(\d+[\.\)]\s*)?", "", p).strip()
            if len(p) > 0:
                cleaned_parts.append(p)

        for crit in cleaned_parts:
            rows.append({"nctid": nctid, "criterion": crit})

    expanded = pd.DataFrame(rows).drop_duplicates().reset_index(drop=True)
    print("Expanded to", len(expanded), "rows")
    return expanded
```

```
criteria_df = load_and_expand_criteria(EXCEL_PATH, crit_col="inclusion")
criteria_df.head()
```

```
Columns: ['nctid', 'inclusion']
Expanded to 26369 rows
```

	nctid	criterion
0	NCT00001137	HIV-1 infected
1	NCT00001137	Enrolled in an AIDS Clinical Trial Group (ACTG...
2	NCT00001137	Willing to provide consent for the release and...
3	NCT00001137	Life expectancy of at least 24 weeks
4	NCT00001137	Parent or guardian willing to provide informed...

Next steps: Generate code with `criteria_df` New interactive sheet

```
# text normalization
def normalize_text(s: str) -> str:
    s = s.lower()
    s = re.sub(r"[^a-z0-9\s]", " ", s)
    s = re.sub(r"\s+", " ", s).strip()
    return s

criteria_df["clean_text"] = criteria_df["criterion"].apply(normalize_text)
criteria_df.head()
```

	nctid	criterion	clean_text
0	NCT00001137	HIV-1 infected	hiv 1 infected
1	NCT00001137	Enrolled in an AIDS Clinical Trial Group (ACTG...	enrolled in an aids clinical trial group actg ...
2	NCT00001137	Willing to provide consent for the release and...	willing to provide consent for the release and...
3	NCT00001137	Life expectancy of at least 24 weeks	life expectancy of at least 24 weeks
4	NCT00001137	Parent or guardian willing to provide informed...	parent or guardian willing to provide informed...

Next steps: Generate code with `criteria_df` New interactive sheet

```
# run a tf-idf + Vectorization
# KMeans Clustering Algorithm
def vectorize_tfidf(texts, max_features=5000, ngram_range=(1, 2)):
    vectorizer = TfidfVectorizer(
        max_features=max_features,
        ngram_range=ngram_range,
        stop_words="english"
    )
    X = vectorizer.fit_transform(texts)
    return X, vectorizer

def cluster_kmeans(X, n_clusters: int, random_state: int = 42):
    model = KMeans(
        n_clusters=n_clusters,
        n_init=10,
        random_state=random_state
    )
    labels = model.fit_predict(X)
    return model, labels


K_TFIDF = 25

X_tfidf, tfidf_vec = vectorize_tfidf(criteria_df["clean_text"].tolist())
kmeans_tfidf, labels_tfidf = cluster_kmeans(X_tfidf, n_clusters=K_TFIDF)

criteria_df["cluster_tfidf"] = labels_tfidf
# now make DBSCAN
criteria_df.head()
```

	nctid	criterion	clean_text	cluster_tfidf
0	NCT00001137	HIV-1 infected	hiv 1 infected	3
1	NCT00001137	Enrolled in an AIDS Clinical Trial Group (ACTG...	enrolled in an aids clinical trial group actg ...	6
2	NCT00001137	Willing to provide consent for the release and...	willing to provide consent for the release and...	6
3	NCT00001137	Life expectancy of at least 24 weeks	life expectancy of at least 24 weeks	5
4	NCT00001137	Parent or guardian willing to provide informed...	parent or guardian willing to provide informed...	7

Next steps: Generate code with `criteria_df` New interactive sheet

Start coding or generate with AI.

```
from sklearn.cluster import DBSCAN

def clusterDBSCAN(X, eps, min_smples):
    model = DBSCAN(eps=eps, min_samples=min_smples, metric='euclidean')
    labels = model.fit_predict(X)
    return model, labels

# run dbscan
eps = 0.5
min_samples = 5
dbscan_model, dbscan_labels = clusterDBSCAN(X_tfidf, eps, min_samples)
newdf = criteria_df.copy()

newdf["cluster_dbscan"] = dbscan_labels

newdf.head(20)
```

	nctid	criterion	clean_text	cluster_tfidf	cluster_dbscan
0	NCT00001137	HIV-1 infected	hiv 1 infected	3	-1
1	NCT00001137	Enrolled in an AIDS Clinical Trial Group (ACTG...	enrolled in an aids clinical trial group actg ...	6	-1
2	NCT00001137	Willing to provide consent for the release and...	willing to provide consent for the release and...	6	-1
3	NCT00001137	Life expectancy of at least 24 weeks	life expectancy of at least 24 weeks	5	-1
4	NCT00001137	Parent or guardian willing to provide informed...	parent or guardian willing to provide informed...	7	-1
5	NCT00001137	Active alcohol or drug abuse that may interfer...	active alcohol or drug abuse that may interfer...	3	-1
6	NCT00003199	Patients with inflammatory (stage IIIb) or res...	patients with inflammatory stage iiib or respo...	4	-1
7	NCT00003199	Patients should have received 4-7 cycles of an...	patients should have received 4 7 cycles of an...	4	-1
8	NCT00003199	Patient has received Cytoxan 4 gm/m\^2 x 1 and...	patient has received cytoxan 4 gm m 2 x 1 and ...	15	-1
9	NCT00003199	Stem cells were collected after mobilization w...	stem cells were collected after mobilization w...	3	-1
10	NCT00003199	The patient must have the capacity to give inf...	the patient must have the capacity to give inf...	7	-1
11	NCT00003199	Hepatic function: Bilirubin =\< 2 mg%; SGOT or...	hepatic function bilirubin 2 mg sgot or sgpt 2...	21	-1
12	NCT00003199	Renal function: Creatinine =\< 2.0 mg/dl or a ...	renal function creatinine 2 0 mg dl or a creat...	21	-1
13	NCT00003199	Pre-Study tests have been performed as	pre study tests have been performed as	6	-1

Next steps: Generate code with `newdf` New interactive sheet

```
print(dbscan_labels[0]*5*-1==5)
```

```
True
```

```
Start coding or generate with AI.
```

```
# Comoute silhouette score for Kmeans
sample_idx = np.random.choice(X_tfidf.shape[0], size=min(5000, X_tfidf.shape[0]), replace=False)
sil_tfidf = silhouette_score(X_tfidf[sample_idx], labels_tfidf[sample_idx])
print("TFIDF KMeans silhouette (sample):", sil_tfidf)




# Compute silhouette score for DBSCAN
sample_idx = np.random.choice(
    X_tfidf.shape[0],
    size=min(5000, X_tfidf.shape[0]),
    replace=False
)

X_sample = X_tfidf[sample_idx]
labels_sample = dbscan_labels[sample_idx]

# Silhouette requires at least 2 non-noise clusters
unique_clusters = [c for c in set(labels_sample) if c != -1]

if len(unique_clusters) >= 2:
    sil_dbscan = silhouette_score(X_sample, labels_sample)
    print("DBSCAN silhouette (sample):", sil_dbscan)
else:
    print("DBSCAN silhouette cannot be computed — fewer than 2 clusters found.")

# in this case  DBSCAN is worse than random assignment, so we ignore it
# This means that we have to optimize it
```

```
TFIDF KMeans silhouette (sample): 0.03811533251679994
DBSCAN silhouette (sample): -0.1776735279698304
```

```
# run sentence embedding to breakdown words into roots
def build_embeddings(texts, model_name: str = "all-MiniLM-L6-v2", batch_size: int = 64):
    model = SentenceTransformer(model_name)
    embeddings = model.encode(
        texts,
        batch_size=batch_size,
        show_progress_bar=True,
        convert_to_numpy=True,
        normalize_embeddings=True
    )
    return embeddings, model

embeddings, sbert_model = build_embeddings(criteria_df["criterion"].tolist())
print("Embedding shape:", embeddings.shape)


# run hierarchial clustering in bottom up (algglomertive clustering)
def cluster_agglomerative(X, n_clusters: int):
    model = AgglomerativeClustering(
        n_clusters=n_clusters,
        linkage="ward"
    )
    labels = model.fit_predict(X)
    return model, labels

K_EMB = 30  # 20,30

agg_model, labels_emb = cluster_agglomerative(embeddings, n_clusters=K_EMB)
criteria_df["cluster_emb"] = labels_emb
sample_idx = np.random.choice(embeddings.shape[0], size=min(5000, embeddings.shape[0]), replace=False)
sil_emb = silhouette_score(embeddings[sample_idx], labels_emb[sample_idx])
print("Embedding Agglomerative silhouette (sample):", sil_emb)
```

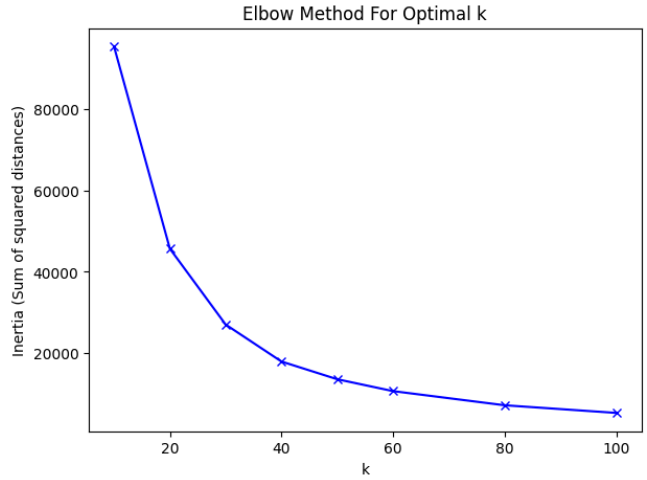
```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
Batches: 100%                                                413/413 [10:30<00:00,  4.46it/s]
Embedding shape: (26369, 384)
Embedding Agglomerative silhouette (sample): 0.062006738
```

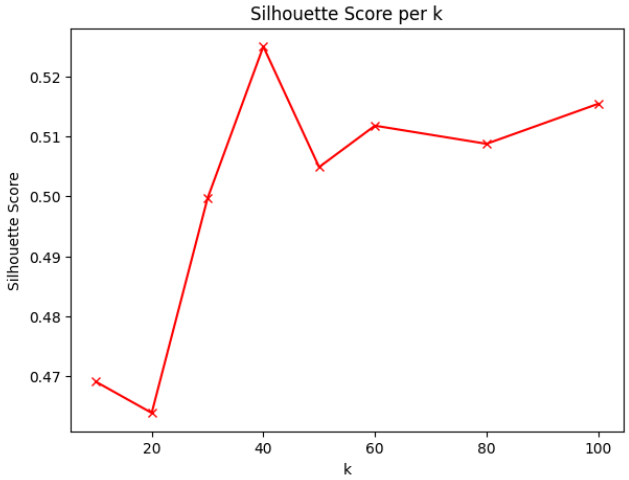
```
import umap
import matplotlib.pyplot as plt

# Optimizing UMAP
# Tighter clusters often come from lower n_neighbors and lower min_dist
reducer = umap.UMAP(
    n_neighbors=15,
    n_components=5,
    min_dist=0.0,
    metric='cosine',
    random_state=42
)

X_umap_optimized = reducer.fit_transform(embeddings)
print("UMAP reduced shape:", X_umap_optimized.shape)
```

```
/usr/local/lib/python3.12/dist-packages/umap/umap_.py:1952: UserWarning: n_jobs value 1 overridden to 1 by setting random_state.
  warn(
UMAP reduced shape: (26369, 5)
```

```
# 2. Optimize hyperparameter based on n_clusters
range_n_clusters = [10, 20, 30, 40, 50, 60, 80, 100] # Adjust range based on dataset size
best_k = 0
best_score = -1
inertias = []
sil_scores = []

print("Optimizing KMeans...")
for n_clusters in range_n_clusters:
    # Use the UMAP reduced data, it usually clusters better than raw SBERT
    clusterer = KMeans(n_clusters=n_clusters, random_state=42, n_init=10)
    cluster_labels = clusterer.fit_predict(X_umap_optimized)

    silhouette_avg = silhouette_score(X_umap_optimized, cluster_labels)
    inertias.append(clusterer.inertia_)
    sil_scores.append(silhouette_avg)

    print(f"For n_clusters = {n_clusters}, Silhouette Score = {silhouette_avg:.4f}")

    if silhouette_avg > best_score:
        best_score = silhouette_avg
        best_k = n_clusters

# Plotting the Elbow and Silhouette
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 5))

# Elbow Plot
ax1.plot(range_n_clusters, inertias, 'bx-')
ax1.set_xlabel('k')
ax1.set_ylabel('Inertia (Sum of squared distances)')
ax1.set_title('Elbow Method For Optimal k')

# Silhouette Plot
ax2.plot(range_n_clusters, sil_scores, 'rx-')
ax2.set_xlabel('k')
ax2.set_ylabel('Silhouette Score')
ax2.set_title('Silhouette Score per k')
plt.show()

print(f"Best K found: {best_k}")
```

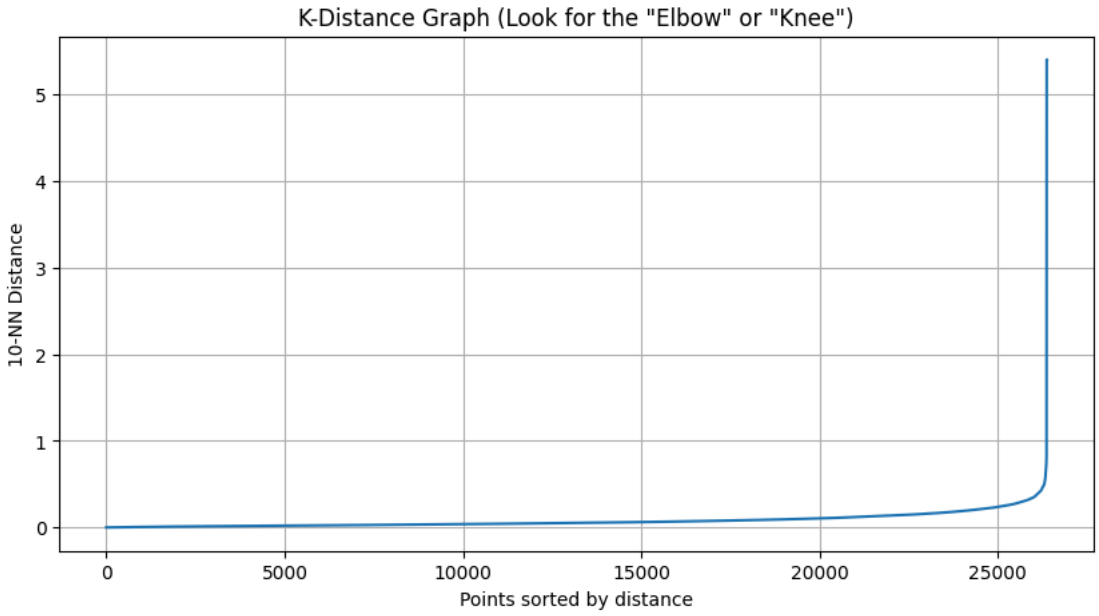
```
Optimizing KMeans...
For n_clusters = 10, Silhouette Score = 0.4690
For n_clusters = 20, Silhouette Score = 0.4638
For n_clusters = 30, Silhouette Score = 0.4997
For n_clusters = 40, Silhouette Score = 0.5250
For n_clusters = 50, Silhouette Score = 0.5049
For n_clusters = 60, Silhouette Score = 0.5118
For n_clusters = 80, Silhouette Score = 0.5088
For n_clusters = 100, Silhouette Score = 0.5155
```






```
Best K found: 40
```

```
from sklearn.neighbors import NearestNeighbors

# 3. FIND OPTIMAL EPS FOR DBSCAN
# We look at the distance to the k-th nearest neighbor (k = 2 * n_components usually works)
k = 10 # Since we have 5 components in UMAP
nbrs = NearestNeighbors(n_neighbors=k).fit(X_umap_optimized)
distances, indices = nbrs.kneighbors(X_umap_optimized)

# Sort distance values by ascending value and plot
distance_desc = sorted(distances[:, k-1], reverse=False)

plt.figure(figsize=(10, 5))
plt.plot(distance_desc)
plt.title('K-Distance Graph (Look for the "Elbow" or "Knee")')
plt.ylabel(f'{k}-NN Distance')
plt.xlabel('Points sorted by distance')
plt.grid(True)
plt.show()

# INSTRUCTION: Look at the Y-axis value where the curve shoots up.
# That Y-value is your optimal 'eps'.
```




```
# --- FINAL CLUSTERING WITH OPTIMIZED PARAMS ---

# 1. Final KMeans
final_kmeans = KMeans(n_clusters=best_k, random_state=42, n_init=10)
criteria_df['cluster_kmeans_opt'] = final_kmeans.fit_predict(X_umap_optimized)

# 2. Final DBSCAN (Replace 0.3 with the value from your K-distance plot)
# min_samples is usually good around 5 to 10 for this data size
final_dbscan = DBSCAN(eps=0.3, min_samples=10, metric='euclidean')
criteria_df['cluster_dbscan_opt'] = final_dbscan.fit_predict(X_umap_optimized)

# Calculate final score (Filtering out noise -1 for DBSCAN)
mask = criteria_df['cluster_dbscan_opt'] != -1
if mask.sum() > 0:
    score = silhouette_score(X_umap_optimized[mask], criteria_df.loc[mask, 'cluster_dbscan_opt'])
    print(f"Final Optimized DBSCAN Silhouette (excluding noise): {score:.4f}")
```

```
Final Optimized DBSCAN Silhouette (excluding noise): 0.0533
```

```
# Optimized DBSCAN application
# Look at the K-Distance graph above.
# Set 'best_eps' to the Y-value where the curve bends sharply (the "elbow").
best_eps = 0.15   # <--- CHANGE THIS based on your graph!

print(f"Running Final DBSCAN with eps={best_eps}...")
final_dbscan = DBSCAN(
    eps=best_eps,
    min_samples=10,
    metric='euclidean'
)

dbscan_umap_labels = final_dbscan.fit_predict(X_umap_optimized)
criteria_df["cluster_dbscan_opt"] = dbscan_umap_labels

print("Clustering complete.")
```

```
Running Final DBSCAN with eps=0.15...
Clustering complete.
```

```
mask = dbscan_labels != -1
X_non_noise = X_umap_optimized[mask]
labels_non_noise = dbscan_labels[mask]

if len(np.unique(labels_non_noise)) >= 2 and len(labels_non_noise) > 50:
    sample_idx = np.random.choice(
        X_non_noise.shape[0],
        size=min(5000, X_non_noise.shape[0]),
        replace=False
    )
```

```
        sil_dbscan_umap = silhouette_score(
            X_non_noise[sample_idx],
            labels_non_noise[sample_idx]
        )
        print("DBSCAN+UMAP silhouette (sample non-noise):", sil_dbscan_umap)
    else:
        print("DBSCAN+UMAP silhouette cannot be computed (not enough clusters).")
```

```
DBSCAN+UMAP silhouette (sample non-noise): 0.070811
```

```
from sklearn.cluster import DBSCAN
best_eps = 0.15
best_min_samples = 10
print(f"Generating labels with eps={best_eps}...")
final_dbscan = DBSCAN(
    eps=best_eps,
    min_samples=best_min_samples,
    metric='euclidean'
)
labels = final_dbscan.fit_predict(X_umap_optimized)

criteria_df['cluster_dbscan_umap'] = labels

print("Column 'cluster_dbscan_umap' created successfully.")
print(f"Clusters found: {len(set(labels)) - (1 if -1 in labels else 0)}")

inspect_dbscan_umap(criteria_df, n_examples=5)
```

```
================================================================================
```

```
DBSCAN+UMAP Cluster = 247
- [NCT01689519] Adequate hematologic and end organ function
- [NCT01765556] Adequate hematologic and end organ function
- [NCT01849666] Adequate hematologic and end organ function
```

```
def inspect_dbscan_umap(df, label_col="cluster_dbscan_umap", n_examples=5):
    for label in sorted(df[label_col].unique()):
        if label == -1:   # skip noise
            continue
        subset = df[df[label_col] == label].head(n_examples)
        print("="*80)
        print(f"DBSCAN+UMAP Cluster = {label}")
        for _, row in subset.iterrows():
            print(f"- [{row['nctid']}] {row['criterion']}")
        print()

inspect_dbscan_umap(criteria_df, n_examples=5)
```

```
- [NCT01820364] confirmed BRAF V600 mutation
- [NCT01910181] Positive BRAF V600 mutation result determined by a designated laboratory using the Cobas 4800 BRAF V600 Mutatio
- [NCT02133222] Known genotype BRAF V600

================================================================================
DBSCAN+UMAP Cluster = 241
- [NCT01233505] First- or second-line metastatic colorectal cancer
- [NCT01291420] Colorectal tumors
- [NCT01315990] Histologically-confirmed metastatic colorectal cancer (primary tumor or metastasis)
- [NCT01372527] Colorectal cancer: Women and men 52-75 years old
- [NCT02704832] Colon and rectum : metastatic (unresectable metastasis),

================================================================================
DBSCAN+UMAP Cluster = 242
- [NCT01240213] No menopausal HRT use of any type including vaginal X 6 months and willing to avoid use for study duration
- [NCT01381445] A female subject is eligible to participate if she is of: • Non-childbearing potential defined as pre-menopausa
- [NCT01899703] A female subject is eligible to participate if she is not pregnant, as confirmed by a negative serum human cho
- [NCT02224781] STEP 2 (CROSSOVER ARMS): Women must not be pregnant or breast-feeding, as the effects of ipilimumab + nivolumab
- [NCT02466152] A woman is eligible to participate if she is of non-reproductive potential, defined as: 1. Postmenopausal (inc

================================================================================
DBSCAN+UMAP Cluster = 243
- [NCT01256489] Bilateral legal blindness (\<20/200 in better eye)
- [NCT01256489] Able to administer eye medications or have a care giver able and willing to do same
- [NCT01467310] History or current evidence/risk of retinal vein occlusion (RVO) or central serous retinopathy (CSR):
- [NCT01838655] Bilateral visual acuity E-ETDRS EVA letter score of less than or equal to 83 (i.e., Snellen equivalent of 20/2
- [NCT01838655] Bilateral iris transillumination that can be seen in clinical photographs. 2. Predominant contralateral decussa

================================================================================
DBSCAN+UMAP Cluster = 244
- [NCT01276704] Ki-67 ≥2% positivity (≥500 cells).
- [NCT01583426] \- cT1c and Ki67 \> 20%
- [NCT01669265] Ki67 index
- [NCT04065321] proliferation index Ki-67 \< 20%;
- [NCT04134598] Ki67 ≤20% by IHC staining;

================================================================================
DBSCAN+UMAP Cluster = 245
- [NCT01419184] Less than 24 hours post hospital admission
- [NCT01965444] Subject is admitted to an intensive Care Unit (ICU) and has an anticipated ICU course of 48 hours or greater.
- [NCT01965444] Subject is anticipated to be bedridden for ore than 6 hours per day.
- [NCT02152358] mechanical ventilation \> 96 hrs and expected duration of mechanical ventilation of at least 2 days
- [NCT02325388] Expected to have a length of stay on the unit of at least three days.

================================================================================
DBSCAN+UMAP Cluster = 246
- [NCT00802672] Study participant or legal guardian was willing and able to read and sign an IRB approved ICF, which included a
- [NCT01543126] Got ICF before enrollment;
- [NCT01679197] If ≥ 18 years of age, is able to read, understand and sign the U of M IRBMED approved informed consent form (IC
- [NCT02209662] Prior to the first clinical intervention, a signed Informed Consent Form (ICF) and Data Consent Form (DCF) must
- [NCT02363842] Subject, or their legal representative, is able and willing to sign an informed consent form (ICF) and willing

================================================================================
DBSCAN+UMAP Cluster = 247
- [NCT01689519] Adequate hematologic and end organ function
- [NCT01765556] Adequate hematologic and end organ function
- [NCT01849666] Adequate hematologic and end organ function
- [NCT01973309] Adequate hematologic and end organ function
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
def inspect_clusters(df: pd.DataFrame,
                     label_col: str,
                     n_examples: int = 5):
    for label in sorted(df[label_col].unique()):
        subset = df[df[label_col] == label].head(n_examples)
        print("=" * 80)
        print(f"Cluster {label_col} = {label}  (showing {len(subset)} examples)")
        for _, row in subset.iterrows():
            print(f"- [{row['nctid']}] {row['criterion']}")
        print()

# Example: quickly look at a few clusters in each method
inspect_clusters(criteria_df, "cluster_tfidf", n_examples=4)
inspect_clusters(criteria_df, "cluster_emb", n_examples=4)
print("DBSCAN AND DIM REDUCTION WITH UMAP")
inspect_clusters(criteria_df, "cluster_dbscan_umap", n_examples=5)
```

```
- [NCT01149083] Female, age \>= 18 years. Because no dosing or adverse event data are currently available on the use of ABT-888
- [NCT01246102] Age greater than or equal to 18 years. Because no dosing or adverse event data are currently available on the u
- [NCT01894451] Age ≥ 21 years. Because no dosing or adverse event data are currently available on the use of 89Zr-bevacizumab

================================================================================
Cluster cluster_dbscan_umap = 236  (showing 5 examples)
- [NCT01010854] absolute neutrophil count \>1,500/mcL
- [NCT01029925] Absolute neutrophil count \>1,500/mcL
- [NCT01575522] Absolute neutrophil count \>= 1,500/mcL
- [NCT01705340] Absolute neutrophil count \>= 1,000/mcL
- [NCT01727076] Absolute neutrophil count \> 1,000/mcL

================================================================================
Cluster cluster_dbscan_umap = 237  (showing 5 examples)
- [NCT00324259] Patients with ER+ HER2+ disease are eligible even of they have received trastuzumab in the past (and even if it
- [NCT00526045] Received up to 3 prior anti HER2 based regimens (i.e. trastuzumab and/or lapatinib in combination with other ag
- [NCT00526045] Patients who develop metastases while receiving adjuvant or neo-adjuvant trastuzumab are eligible. HER2 positiv
- [NCT00704158] Subjects with HER2/neu overexpressing tumors must have been treated with trastuzumab except in situations where
- [NCT01029925] HER-2 positive breast cancer should have received Trastuzumab, in either the adjuvant or metastatic setting.

================================================================================
Cluster cluster_dbscan_umap = 238  (showing 5 examples)
- [NCT01029925] Non-small cell lung cancer patients should have received at least platinum based chemotherapy in the adjuvant,
- [NCT01910844] Patient non previously treated by platinum salts,
- [NCT02210663] Subjects with recurrent high grade serous ovarian cancer who completed or discontinued platinum based therapy;
- [NCT02365662] Expanded Safety Cohort participants must have confirmed metastatic lung cancer and progressed after receiving p
- [NCT02561832] Prior use of platinum compound in the advanced or metastatic setting. Previous exposure to platinum compounds

================================================================================
Cluster cluster_dbscan_umap = 239  (showing 5 examples)
- [NCT01095848] Patients with prostate cancer who have failed at least 1 course of an accepted hormonal therapy. Specifically p
- [NCT01522820] Prostate cancer: patients with metastatic, castrate refractory prostate cancer; the use of luteinizing hormone
- [NCT02704832] Prostate cancer : metastatic and refractory to hormonal castration,
- [NCT03218826] DISEASE SPECIFIC EXPANSION COHORTS: Prostate cancers patients enrolled on this study (applies to all prostate c
- [NCT03310541] Prostate Cancer Clinical Trials Working Group 3 (PCWG3) criteria.

================================================================================
Cluster cluster_dbscan_umap = 240  (showing 5 examples)
- [NCT01164891] Positive BRAF V600E mutation result (by Roche CoDx test)
- [NCT01705392] Known BRAF mutation
- [NCT01820364] confirmed BRAF V600 mutation
- [NCT01910181] Positive BRAF V600 mutation result determined by a designated laboratory using the Cobas 4800 BRAF V600 Mutatio
- [NCT02133222] Known genotype BRAF V600

================================================================================
Cluster cluster_dbscan_umap = 241  (showing 5 examples)
- [NCT01233505] First- or second-line metastatic colorectal cancer
- [NCT01291420] Colorectal tumors
- [NCT01315990] Histologically-confirmed metastatic colorectal cancer (primary tumor or metastasis)
- [NCT01372527] Colorectal cancer: Women and men 52-75 years old
- [NCT02704832] Colon and rectum : metastatic (unresectable metastasis),

================================================================================
Cluster cluster_dbscan_umap = 242  (showing 5 examples)
- [NCT01240213] No menopausal HRT use of any type including vaginal X 6 months and willing to avoid use for study duration
- [NCT01381445] A female subject is eligible to participate if she is of: • Non-childbearing potential defined as pre-menopausa
- [NCT01899703] A female subject is eligible to participate if she is not pregnant, as confirmed by a negative serum human cho
- [NCT02224781] STEP 2 (CROSSOVER ARMS): Women must not be pregnant or breast-feeding, as the effects of ipilimumab + nivolumab
```

```
# Map numbers to string labels
# Look at all clustering algorithms and see which ones are better
```

```
def numeric_to_string_labels(int_labels: np.ndarray) -> np.ndarray:
    uniq = np.unique(int_labels)
    mapping = {c: f"label{chr(ord('A') + i)}" for i, c in enumerate(uniq)}
    return np.array([mapping[c] for c in int_labels])

criteria_df["cluster_final"] = numeric_to_string_labels(criteria_df["cluster_emb"].values)

results_df = criteria_df[["nctid", "criterion", "cluster_final"]].copy()
results_df.columns = ["nctid", "criterion", "cluster"]


RESULTS_PATH = "results_sendroff_partner.txt"

results_df.to_csv(
    RESULTS_PATH,
    sep="\t",
    index=False
)

print("Saved results to", RESULTS_PATH)
results_df.head()
```

```
Saved results to results_sendroff_partner.txt
```

	nctid	criterion	cluster
0	NCT00001137	HIV-1 infected	labelK
1	NCT00001137	Enrolled in an AIDS Clinical Trial Group (ACTG...	labelE
2	NCT00001137	Willing to provide consent for the release and...	labelE
3	NCT00001137	Life expectancy of at least 24 weeks	label[
4	NCT00001137	Parent or guardian willing to provide informed...	labelH

Next steps: Generate code with `results_df` New interactive sheet

```
import numpy as np

# Helper function to map numbers to strings, handling Noise (-1)
def get_string_label(label):
    if label == -1:
        return "labelNoise"
    # Maps 0 -> labelA, 1 -> labelB, etc.
    # Using modulo 26 to wrap around if you have >26 clusters (labelA...labelZ, labelA...)
    letter = chr(ord('A') + (label % 26))
    suffix = str(label // 26) if label >= 26 else ""
    return f"label{letter}{suffix}"

# Apply to the Optimized DBSCAN column (or KMeans if that score was better!)
# Assuming DBSCAN was your best approach:
criteria_df["final_string_label"] = criteria_df["cluster_dbscan_opt"].apply(get_string_label)

# Create the final dataframe for export
final_results = criteria_df[["nctid", "criterion", "final_string_label"]].copy()
final_results.columns = ["nctid", "criterion", "cluster"]

# Save to .txt as requested
output_filename = "results_sendroff_partner.txt"
final_results.to_csv(output_filename, sep="\t", index=False)

print(f"Success! Saved {len(final_results)} rows to {output_filename}")
print(final_results.head())
```

```
Success! Saved 26369 rows to results_sendroff_partner.txt
         nctid                                          criterion     cluster
```