1. **What kind of data would you like to work with for your project? List any potential sources.**

*Topic:* Skin Cancer Classification using Topological Data Analysis

The project is focused on using topological data analysis (TDA), specifically persistent homology,in oncology. The data that would be ideal for this project would be patient-specific data related to cancer treatment responses, clinical outcomes, disease classification and architecture in cancer. Potential sources of this data will include MNIST data.

Link:https://www.kaggle.com/datasets/kmader/skin-cancer-mnist-ham10000

1. **What kind of questions would you hope to answer?**

TDA has been successfully applied in a variety of medical context. The questions that I will try to answer are What is topological data analysis and how does it help in analyzing biomedical data, successful applications of TDA in the field of oncology, I will also try to find if TDA can be used to analyze cancer time-series data. Benefits and Limitations of Persistent Homology

1. **What methods from TDA would be applicable? This includes both theoretical frameworks and existing software.**
2. Persistent Homology
3. Computational Homology
4. GUDHI
5. Python Mapper
6. **Are there similar studies that have been already published?**

No (not sure)

1. **Is your project complementary to those of other students in the class? While there will not be projects with a group submission, it's fine if some of you work on related questions/data.**

No

I have also tried to implement distance matrix on our dataset. Since the datasets contain different types of information, we need to clarify the specific use and purpose for whicg we would like to compute the distance matrix. But we don’t have any common column in our given two datasets as one datset contains information about skin lesion images and the other dataset consists if pixel values of RGB images. So to implement distance I have used the 'age' column from the "HAM10000\\_metadata.csv" dataset as a representative numeric feature. We then convert the 'age' column to numeric representation using pd.to\_numeric() with the errors='coerce' parameter to handle any non-numeric values.

# Split the data into training and testing sets

y = data2['label'].to\_numpy()

X\_train\_topo, X\_test\_topo, y\_train, y\_test = train\_test\_split(X\_topo, y, test\_size=0.2, random\_state=42)

# Create logistic regression model

model = LogisticRegression()

# Train model on topological features

model.fit(X\_train\_topo, y\_train)

# Make predictions on testing set

y\_pred = model.predict(X\_test\_topo)

# Calculate accuracy score

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

CNN

import numpy as np

import pandas as pd

import PIL.Image as Image

from sklearn.cluster import KMeans

from sklearn.tree import DecisionTreeClassifier

from sklearn.preprocessing import OneHotEncoder, StandardScaler

from sklearn.model\_selection import train\_test\_split

import tensorflow as tf

from tensorflow.keras import layers

from tensorflow.keras.models import Sequential

df1 = np.array(data2)

np.random.shuffle(df1)

X = df1[:, :-1]

X = X.reshape(X.shape[0], 28, 28, 3) / 255 # Scaling seems to improve accuracy

y = df1[:, -1]

y = y.reshape(y.shape[0], 1)

onehot = OneHotEncoder()

y = onehot.fit\_transform(y).toarray()

X\_train\_topo, X\_test\_topo, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

tensor\_train = tf.convert\_to\_tensor(X\_train\_topo, dtype=tf.float32)

tensor\_test = tf.convert\_to\_tensor(X\_test\_topo, dtype=tf.float32)

model = Sequential()

#conv2d = tf.keras.layers.Conv2D(10, 3, activation="relu", input\_shape=(10, 28, 28, 3))

model.add(layers.Conv2D(80, (5, 5), activation="relu", input\_shape=(28, 28, 3)))

model.add(layers.MaxPooling2D(pool\_size=(2, 2)))

#model.add(layers.BatchNormalization(axis=-1))

model.add(layers.Conv2D(64, (5, 5), activation="relu"))

model.add(layers.MaxPooling2D(pool\_size=(2, 2)))

model.add(layers.Conv2D(64, (4, 4), activation="relu"))

model.add(layers.Flatten())

model.add(layers.Dense(7, activation="softmax", use\_bias=True))

model.summary()

model.compile(optimizer="adam", loss="categorical\_crossentropy", metrics=["accuracy"])

checkpoint\_callback = tf.keras.callbacks.ModelCheckpoint(

    filepath=".",

    monitor="val\_accuracy",

    mode="max",

    save\_best\_only=True

)

history = model.fit(

    x=tensor\_train,

    y=y\_train,

    epochs=125,

    batch\_size=50,

    validation\_split=0.1,

    callbacks=[checkpoint\_callback],

)

print(model.evaluate(X\_test\_topo, y\_test))