CS156 - Pipeline - First Draft

Which Video Will I Like?

Problem Definition

The objective of this project is to predict whether I will "like" a YouTube video based solely on its metadata before watching it. By analyzing patterns in the metadata of videos I have previously liked versus those I have watched but not liked, we aim to build a machine learning model that can forecast my likelihood of liking new videos.

Motivation

With the vast amount of content available on YouTube, personalized recommendations can significantly enhance the viewing experience. Predicting which videos I am likely to like can help in:

- **Streamlining Content Consumption**: Reducing time spent searching for engaging videos.
- **Personal Insights**: Understanding my own viewing preferences and behaviors.
- **Enhancing Recommendation Systems**: Providing a framework that could be adapted for broader recommendation engines.

Objectives

- **Data Acquisition**: Collect metadata from a balanced dataset of liked and unliked videos.
- **Feature Extraction**: Identify and extract relevant metadata features that may influence liking behavior.
- **Model Development**: Train machine learning models to predict the likelihood of liking a video.
- **Evaluation**: Assess model performance using appropriate metrics.
- **Analysis**: Interpret the results to understand key factors influencing the predictions.

Data Collection

To build a robust predictive model, we will collect data from two categories:

1. **Liked Videos (Positive Class)**:
   - **Source**: Videos on which I have pressed the "like" button.
   - **Quantity**: 200 videos.
   - **Method**: Use the YouTube Data API to retrieve metadata of liked videos.
2. **Unliked Videos (Negative Class)**:
   - **Source**: Videos I have watched but did not press the "like" button.
   - **Quantity**: Randomly select 200 videos from a pool of 1,000 recently watched unliked videos.
   - **Method**:

    - Extract watch history from `watch-history.html` obtained via Google Takeout.
    - Filter out videos that have been liked.
    - Randomly select 200 unliked videos to match the number of liked videos.

Feature Extraction

Relevant metadata features to be extracted include:

- **Video Attributes**:
  - Title
  - Description
  - Tags
  - Category ID
  - Duration
  - Default Audio Language
  - Topic Categories
- **Engagement Metrics**:
  - View Count
  - Like Count
  - Comment Count
- **Content Details**:
  - Published Date and Time
  - Content Rating
  - Definition (HD or SD)

Data Preprocessing

To prepare the data for modeling, the following preprocessing steps will be applied:

- **Data Cleaning**:
  - Handle missing or null values.
  - Remove duplicates.
- **Feature Engineering**:
  - Convert duration to total seconds.
  - Extract textual features from title and description (e.g., word counts, sentiment scores).
  - Categorize view counts, like counts, and comment counts into bins.
- **Encoding Categorical Variables**:
  - Use one-hot encoding for categorical features like category ID and language.
  - Encode textual features using techniques like TF-IDF vectors for titles and descriptions.
- **Normalization and Scaling**:
  - Scale numerical features to ensure uniformity.

Modeling

Multiple machine learning algorithms will be explored to find the best predictive model:

- **Baseline Models**:
  - Logistic Regression
  - Decision Trees
- **Advanced Models**:
  - Random Forests
  - Gradient Boosting Machines (e.g., XGBoost, LightGBM)
  - Support Vector Machines
  - Neural Networks (for more complex patterns)

Evaluation

Models will be evaluated using cross-validation and the following performance metrics:

- **Accuracy**: Overall correctness of the model.
- **Precision**: Correctly predicted likes out of all predicted likes.
- **Recall**: Correctly predicted likes out of all actual likes.
- **F1-Score**: Harmonic mean of precision and recall.
- **Confusion Matrix**: To visualize true vs. predicted classifications.
- **ROC Curve and AUC Score**: To evaluate the model's ability to discriminate between classes.

Expected Challenges

- **Imbalanced Data**: Ensuring the dataset remains balanced to prevent bias.
- **Feature Selection**: Identifying which metadata features are most predictive.
- **Overfitting**: Avoiding models that perform well on training data but poorly on unseen data.
- **Data Privacy**: Handling personal data securely and ethically.

Results Interpretation

After training and evaluating the models, we will:

- **Identify Key Predictors**: Determine which features most influence the likelihood of liking a video.
- **Model Comparison**: Compare the performance of different algorithms to select the best model.
- **Practical Implications**: Discuss how the model could be used in real-world scenarios, such as enhancing YouTube's recommendation system.

Conclusion

This project aims to create a personalized predictive model for liking YouTube videos based on metadata. Successful completion could lead to improved content recommendations and a deeper understanding of personal viewing habits.

Future Work

- **Expand Feature Set**: Incorporate additional metadata or user interaction features.
- **Time-Series Analysis**: Analyze how preferences change over time.
- **Apply to Other Users**: Test the model's applicability to predict likes for other users.

Section 2: Converting data into Python dataframe

```
import google_auth_oauthlib.flow
import googleapiclient.discovery
import googleapiclient.errors

# Set up OAuth 2.0 authentication flow
scopes = ["https://www.googleapis.com/auth/youtube.force-ssl"]
client_secrets_file = "../../../Desktop/client_secret.json"

# Create an OAuth flow and get credentials using the local server
method
flow =
google_auth_oauthlib.flow.InstalledAppFlow.from_client_secrets_file(
    client_secrets_file, scopes)

# Use run_local_server() to initiate the OAuth flow
flow.redirect_uri = 'http://localhost:5001/oauth2callback'
credentials = flow.run_local_server(port=5001)

# Create a YouTube client
youtube = googleapiclient.discovery.build("youtube", "v3",
credentials=credentials)


# Request liked videos
request = youtube.videos().list(
    part="snippet,contentDetails",
    myRating="like"
)
response = request.execute()

# Process and print video details
for item in response.get("items", []):
    title = item["snippet"]["title"]
    video_id = item["id"]
    print(f"Title: {title}, Video ID: {video_id}")
```

```
Please visit this URL to authorize this application:
https://accounts.google.com/o/oauth2/auth?
response_type=code&client_id=217986578119-
0l6jjn9u267th7o7rh7bm2uqffaqqq96.apps.googleusercontent.com&redirect_u
ri=http%3A%2F%2Flocalhost%3A5001%2F&scope=https%3A%2F
%2Fwww.googleapis.com%2Fauth%2Fyoutube.force-
ssl&state=rf8n0Btx7nFq7Is7dG9GrKRC83spM3&access_type=offline
```

```
Title:    , 100%      , Video ID: aV1GcmMh-ME
Title: What Is Dynamic Programming and How To Use It, Video ID:
vYquumk4nWw
Title: 8 vs. 8 soccer: Tactics, Formation, Position (3-3-1), Video ID:
iUPthkBfjJM
Title: 🐨      , Video ID: JzEt1rvl8EA
Title:          , Video ID: F4pXf_KIYu4
```

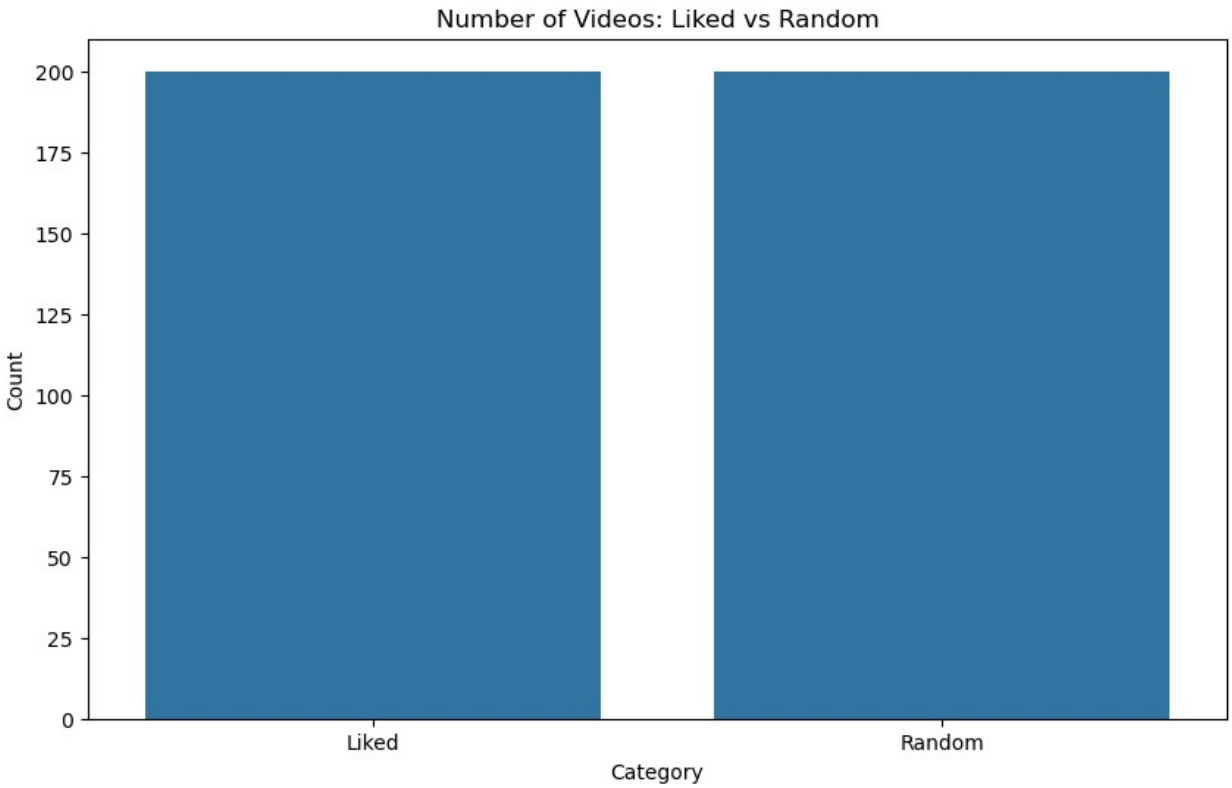
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

def get_liked_videos(max_results=200):
    videos = []
    next_page_token = None

    while len(videos) < max_results:
        request = youtube.videos().list(
            part="snippet,contentDetails,statistics,topicDetails",
            myRating="like",
            maxResults=50,
            pageToken=next_page_token
        )

        response = request.execute()
        videos.extend(response.get("items", []))

        next_page_token = response.get("nextPageToken")
        if not next_page_token:
            break

    return videos[:max_results]

def get_random_videos_details(video_ids):
    videos = []
    for i in range(0, len(video_ids), 50):
        request = youtube.videos().list(
            part="snippet,contentDetails,statistics,topicDetails",
            id=','.join(video_ids[i:i+50])
        )
        response = request.execute()
        videos.extend(response.get("items", []))
    return videos

def get_random_videos(max_results=200, regions=['US', 'KR']):
    search_terms = ["music", "technology", "news", "sports", "gaming",
"cooking", "travel", "science"]
    video_ids = []

    for term in search_terms:
```

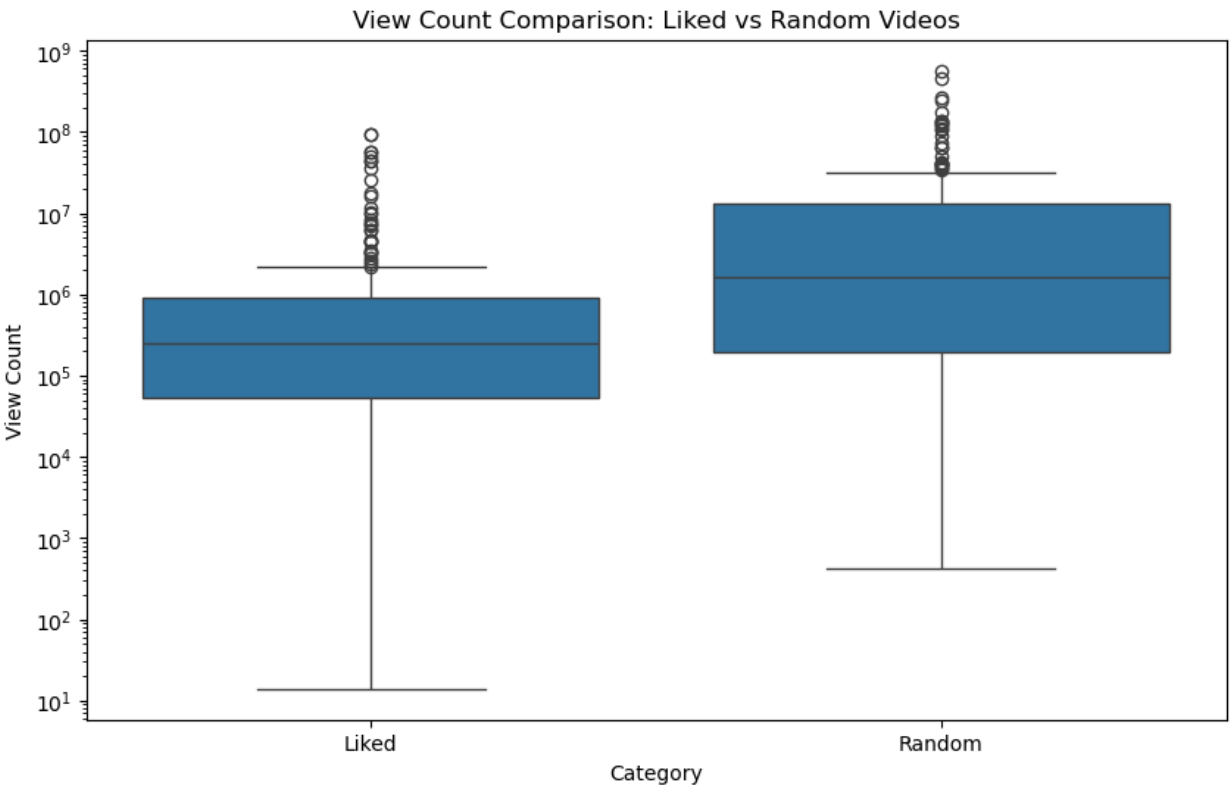
```
        if len(video_ids) >= max_results:
            break

        for region in regions:
            request = youtube.search().list(
                part="snippet",
                q=term,
                type="video",
                maxResults=25,  # Reduce max results to balance
between regions
                regionCode=region
            )

            response = request.execute()
            video_ids.extend([item['id']['videoId'] for item in
response.get("items", [])])

            if len(video_ids) >= max_results:
                break

    return get_random_videos_details(video_ids[:max_results])

# Get 200 liked videos
liked_videos = get_liked_videos(max_results=200)

# Get 200 random videos from YouTube (from US and Korea)
random_videos = get_random_videos(max_results=200, regions=['US',
'KR'])

# Create DataFrame to store video information
def create_dataframe(video_items, category):
    data = []
    for item in video_items:
        snippet = item.get("snippet", {})
        content_details = item.get("contentDetails", {})
        statistics = item.get("statistics", {})
        topic_details = item.get("topicDetails", {})

        video_info = {
            "category": category,
            "title": snippet.get("title", "N/A"),
            "description": snippet.get("description", "N/A"),
            "tags": snippet.get("tags", "N/A"),
            "category_id": snippet.get("categoryId", "N/A"),
            "duration": content_details.get("duration", "N/A"),
            "view_count": int(statistics.get("viewCount", 0)),
            "like_count": int(statistics.get("likeCount", 0)),
            "comment_count": int(statistics.get("commentCount", 0)),
            "topic_categories": topic_details.get("topicCategories",
"N/A"),
```

```
            "language": snippet.get("defaultAudioLanguage",
snippet.get("defaultLanguage", "N/A"))
        }
        data.append(video_info)
    return pd.DataFrame(data)

# Create DataFrames for liked and random videos
liked_videos_df = create_dataframe(liked_videos, "Liked")
random_videos_df = create_dataframe(random_videos, "Random")

# Combine both DataFrames
combined_df = pd.concat([liked_videos_df, random_videos_df],
ignore_index=True)

# Save DataFrame to CSV
combined_df.to_csv("combined_videos.csv", index=False)

# Display the DataFrame
print(combined_df.head())

# Summarize the dataset
print("\nDataset Summary:")
print(combined_df.describe())

# Bar graph to compare number of liked and random videos
plt.figure(figsize=(10, 6))
sns.countplot(x="category", data=combined_df)
plt.title("Number of Videos: Liked vs Random")
plt.xlabel("Category")
plt.ylabel("Count")
plt.show()

# Compare views of liked and random videos
plt.figure(figsize=(10, 6))
sns.boxplot(x="category", y="view_count", data=combined_df)
plt.title("View Count Comparison: Liked vs Random Videos")
plt.xlabel("Category")
plt.ylabel("View Count")
plt.yscale("log")  # Use log scale to handle wide range of view counts
plt.show()
```

```
  category                                                  title  \
0    Liked                                , 100%
1    Liked      What Is Dynamic Programming and How To Use It
2    Liked  8 vs. 8 soccer: Tactics, Formation, Position (...
3    Liked                                  🐨
4    Liked

                                            description  \
0  #  #  #  #   #  #  \n...
```

```
1  **Dynamic Programming Tutorial**\nThis is a qu...
2  This video looks at three of the basics for an...
3
4  # #  #  # # #  # \n#Sofa4844...

                                                   tags category_id
duration  \
0  [#  , #  , #  , #  , #  , #  ...          26  PT14M11S
1  [dynamic programming tutorial, dynamic program...          27
PT14M28S
2  [soocer, U9, coaching, 3-3-1- formation, kid-f...          22
PT7M20S
3                                                   N/A          22
PT28S
4  [Sofa4844, Shorts,  ,  ,  ,  ,  ,  ...          10     PT29S

   view_count  like_count  comment_count  \
0         132          16              7
1     1606355       41524           1619
2       85312        1167             23
3       58832         986             15
4     1582811       28947           1270

                                    topic_categories language
0          [https://en.wikipedia.org/wiki/Knowledge]       ko
1          [https://en.wikipedia.org/wiki/Knowledge]       en
2  [https://en.wikipedia.org/wiki/Association_foo...      N/A
3               [https://en.wikipedia.org/wiki/Food]      N/A
4  [https://en.wikipedia.org/wiki/Music, https://...       ko

Dataset Summary:
         view_count    like_count  comment_count
count  4.000000e+02  4.000000e+02     400.000000
mean   1.189981e+07  2.467796e+05    3684.720000
std    4.541627e+07  9.671951e+05   12502.592978
min    1.400000e+01  0.000000e+00       0.000000
25%    9.369200e+04  2.062750e+03      42.000000
50%    6.158440e+05  1.080750e+04     319.500000
75%    4.381516e+06  7.770100e+04    1763.500000
max    5.690540e+08  1.450842e+07  125802.000000
```


Number of Videos: Liked vs Random
Count
200
175
150
125
100
75
50
25
0
Liked
Random
Category


View Count Comparison: Liked vs Random Videos
View Count
10^9
10^8
10^7
10^6
10^5
10^4
10^3
10^2
10^1
Liked
Random
Category

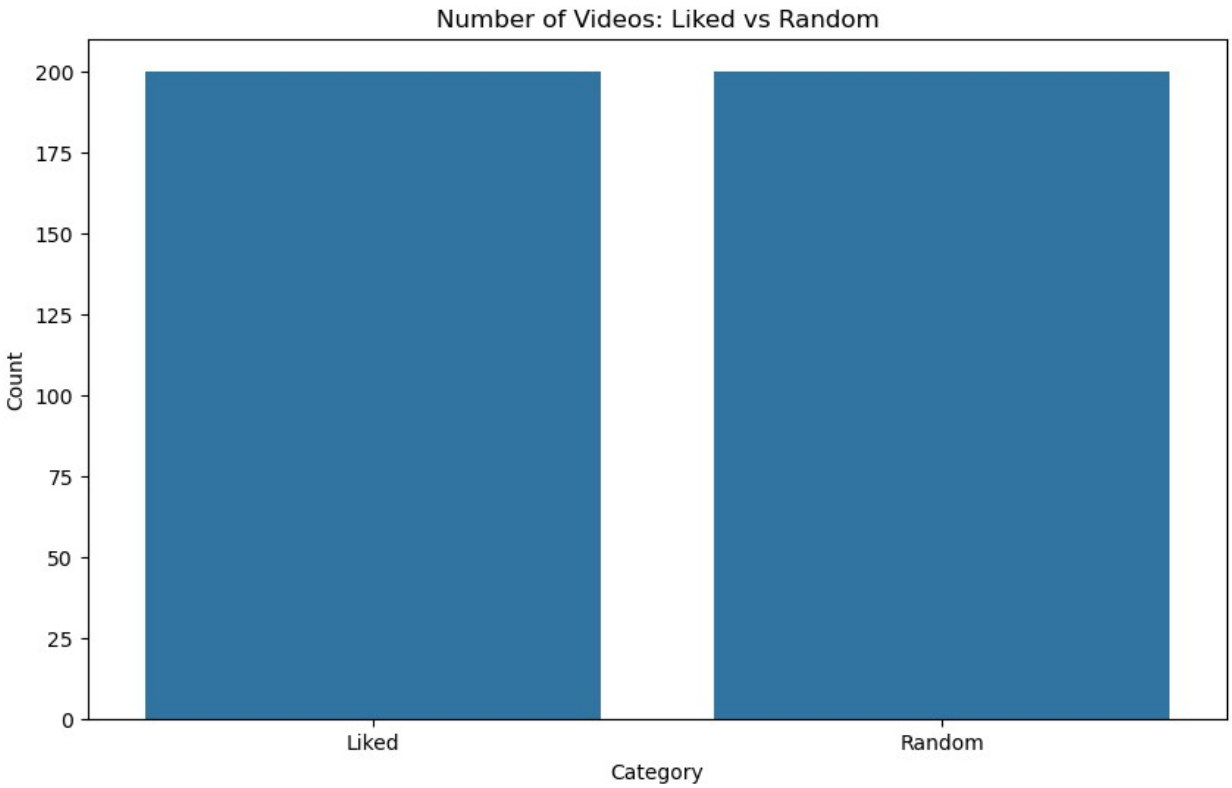
```python
import pandas as pd
from datetime import datetime
import matplotlib.pyplot as plt
import seaborn as sns


def get_liked_videos(max_results=200):
    videos = []
    next_page_token = None

    while len(videos) < max_results:
        request = youtube.videos().list(
            part="snippet,contentDetails,statistics,topicDetails",
            myRating="like",
            maxResults=50,
            pageToken=next_page_token
        )

        response = request.execute()
        videos.extend(response.get("items", []))

        next_page_token = response.get("nextPageToken")
        if not next_page_token:
            break

    return videos[:max_results]

def get_random_videos_details(video_ids):
    videos = []
    for i in range(0, len(video_ids), 50):
        request = youtube.videos().list(
            part="snippet,contentDetails,statistics,topicDetails",
            id=','.join(video_ids[i:i+50])
        )
        response = request.execute()
        videos.extend(response.get("items", []))
    return videos

def get_random_videos(max_results=200, regions=['US', 'KR']):
    video_ids = []

    for region in regions:
        while len(video_ids) < max_results:
            request = youtube.videos().list(
                part="snippet",
                chart="mostPopular",
                regionCode=region,
                maxResults=50
            )
```

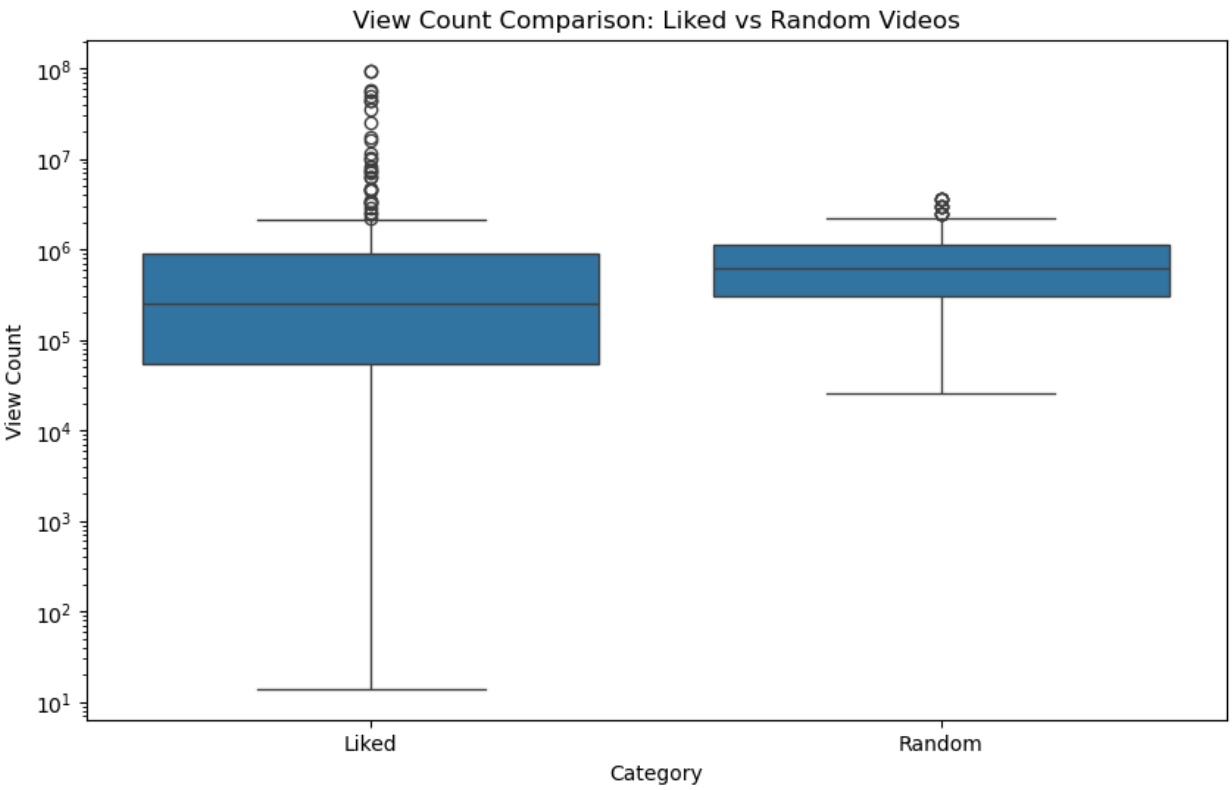
```
            response = request.execute()
            video_ids.extend([item['id'] for item in
response.get("items", [])])

            if len(video_ids) >= max_results:
                break

    return get_random_videos_details(video_ids[:max_results])

# Get 200 liked videos
liked_videos = get_liked_videos(max_results=200)

# Get 200 random videos from YouTube (from US and Korea)
random_videos = get_random_videos(max_results=200, regions=['US',
'KR'])

# Create DataFrame to store video information
def create_dataframe(video_items, category):
    data = []
    for item in video_items:
        snippet = item.get("snippet", {})
        content_details = item.get("contentDetails", {})
        statistics = item.get("statistics", {})
        topic_details = item.get("topicDetails", {})

        video_info = {
            "category": category,
            "title": snippet.get("title", "N/A"),
            "description": snippet.get("description", "N/A"),
            "tags": snippet.get("tags", "N/A"),
            "category_id": snippet.get("categoryId", "N/A"),
            "duration": content_details.get("duration", "N/A"),
            "view_count": int(statistics.get("viewCount", 0)),
            "like_count": int(statistics.get("likeCount", 0)),
            "comment_count": int(statistics.get("commentCount", 0)),
            "topic_categories": topic_details.get("topicCategories",
"N/A"),
            "language": snippet.get("defaultAudioLanguage",
snippet.get("defaultLanguage", "N/A"))
        }
        data.append(video_info)
    return pd.DataFrame(data)

# Create DataFrames for liked and random videos
liked_videos_df = create_dataframe(liked_videos, "Liked")
random_videos_df = create_dataframe(random_videos, "Random")

# Combine both DataFrames
combined_df = pd.concat([liked_videos_df, random_videos_df],
```

```
ignore_index=True)

# Save DataFrame to CSV
combined_df.to_csv("combined_videos.csv", index=False)

# Display the DataFrame
print(combined_df.head())

# Summarize the dataset
print("\nDataset Summary:")
print(combined_df.describe())

# Bar graph to compare number of liked and random videos
plt.figure(figsize=(10, 6))
sns.countplot(x="category", data=combined_df)
plt.title("Number of Videos: Liked vs Random")
plt.xlabel("Category")
plt.ylabel("Count")
plt.show()

# Compare views of liked and random videos
plt.figure(figsize=(10, 6))
sns.boxplot(x="category", y="view_count", data=combined_df)
plt.title("View Count Comparison: Liked vs Random Videos")
plt.xlabel("Category")
plt.ylabel("View Count")
plt.yscale("log")  # Use log scale to handle wide range of view counts
plt.show()
```

```
  category                                                title  \
0    Liked                            , 100%
1    Liked      What Is Dynamic Programming and How To Use It
2    Liked  8 vs. 8 soccer: Tactics, Formation, Position (...
3    Liked                                  🐨
4    Liked

                                            description  \
0  #  #  #  #   #  #  \n...
1  **Dynamic Programming Tutorial**\nThis is a qu...
2  This video looks at three of the basics for an...
3
4  # #  #  # # # # \n#Sofa4844...

                                                tags category_id
duration  \
0  [#  , #  , #  , #  , #  , # ...          26  PT14M11S
1  [dynamic programming tutorial, dynamic program...          27
PT14M28S
2  [soocer, U9, coaching, 3-3-1- formation, kid-f...          22
PT7M20S
```

```
3                                                    N/A          22
PT28S
4  [Sofa4844, Shorts,  ,  ,  ,  ,  ,  ...          10     PT29S

   view_count  like_count  comment_count  \
0         132          16              7
1     1606378       41525           1619
2       85312        1167             23
3       58832         986             15
4     1582845       28948           1270

                                     topic_categories language
0          [https://en.wikipedia.org/wiki/Knowledge]       ko
1          [https://en.wikipedia.org/wiki/Knowledge]       en
2  [https://en.wikipedia.org/wiki/Association_foo...      N/A
3               [https://en.wikipedia.org/wiki/Food]      N/A
4  [https://en.wikipedia.org/wiki/Music, https://...       ko

Dataset Summary:
         view_count    like_count  comment_count
count  4.000000e+02  4.000000e+02     400.000000
mean   2.184960e+06  6.409394e+04    2478.070000
std    8.986955e+06  2.757538e+05    4892.842976
min    1.400000e+01  0.000000e+00       0.000000
25%    1.253030e+05  3.213750e+03     199.000000
50%    4.916530e+05  1.223200e+04     656.000000
75%    1.022161e+06  3.860300e+04    1984.000000
max    9.430828e+07  3.807253e+06   27790.000000
```


Number of Videos: Liked vs Random
Count
200
175
150
125
100
75
50
25
0
Liked
Random
Category


View Count Comparison: Liked vs Random Videos
View Count
10^8
10^7
10^6
10^5
10^4
10^3
10^2
10^1
Liked
Random
Category

```python
import pandas as pd
from datetime import datetime
import matplotlib.pyplot as plt
import seaborn as sns
from bs4 import BeautifulSoup
import re
import random

def get_liked_videos(max_results=200):
    videos = []
    next_page_token = None

    while len(videos) < max_results:
        request = youtube.videos().list(
            part="snippet,contentDetails,statistics,topicDetails",
            myRating="like",
            maxResults=50,
            pageToken=next_page_token
        )

        response = request.execute()
        videos.extend(response.get("items", []))

        next_page_token = response.get("nextPageToken")
        if not next_page_token:
            break

    return videos[:max_results]

def get_random_videos_details(video_ids):
    videos = []
    for i in range(0, len(video_ids), 50):
        try:
            request = youtube.videos().list(
                part="snippet,contentDetails,statistics,topicDetails",
                id=','.join(video_ids[i:i+50])
            )
            response = request.execute()
            videos.extend(response.get("items", []))
        except Exception as e:
            print(f"An error occurred: {e}")
            continue
    return videos

def get_recently_watched_videos_from_html(file_path,
max_results=1000):
    # Read the HTML file
    with open(file_path, 'r', encoding='utf-8') as f:
        soup = BeautifulSoup(f, 'lxml')
```

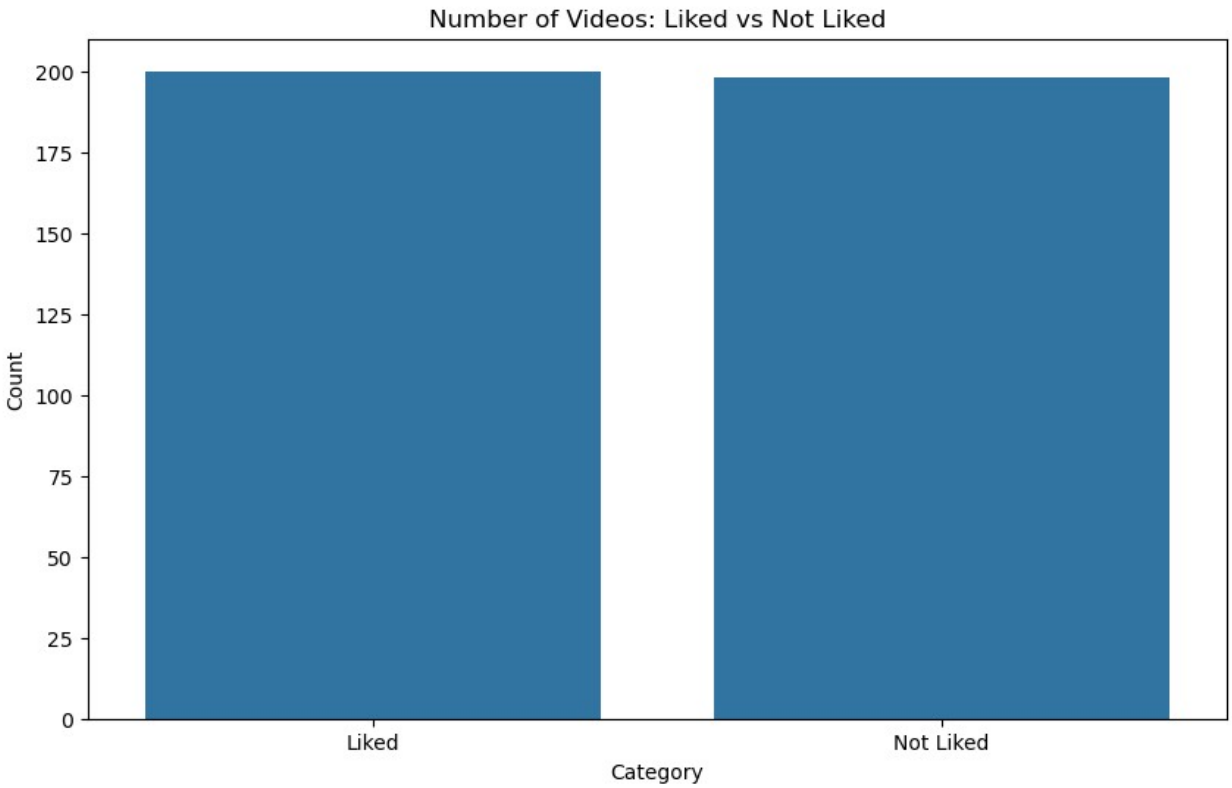
```
    # Find all watch history entries
    entries = soup.find_all('div', class_='mdl-grid')

    video_ids = []
    for entry in entries:
        # Find the anchor tag with the video URL
        a_tag = entry.find('a')
        if a_tag and 'youtube.com/watch' in a_tag.get('href', ''):
            url = a_tag['href']
            # Extract the video ID using regex
            match = re.search(r'v=([^&]+)', url)
            if match:
                video_id = match.group(1)
                video_ids.append(video_id)
                if len(video_ids) >= max_results:
                    break

    # Remove duplicates while preserving order
    video_ids = list(dict.fromkeys(video_ids))

    return video_ids

def create_dataframe(video_items, category):
    data = []
    for item in video_items:
        snippet = item.get("snippet", {})
        content_details = item.get("contentDetails", {})
        statistics = item.get("statistics", {})
        topic_details = item.get("topicDetails", {})

        video_info = {
            "category": category,
            "title": snippet.get("title", "N/A"),
            "description": snippet.get("description", "N/A"),
            "tags": snippet.get("tags", "N/A"),
            "category_id": snippet.get("categoryId", "N/A"),
            "duration": content_details.get("duration", "N/A"),
            "view_count": int(statistics.get("viewCount", 0)),
            "like_count": int(statistics.get("likeCount", 0)),
            "comment_count": int(statistics.get("commentCount", 0)),
            "topic_categories": topic_details.get("topicCategories",
"N/A"),
            "language": snippet.get("defaultAudioLanguage",
snippet.get("defaultLanguage", "N/A"))
        }
        data.append(video_info)
    return pd.DataFrame(data)

# Provide the path to your watch-history.html file
file_path = '/Users/hcoh/Downloads/Takeout 3/YouTube and YouTube
```

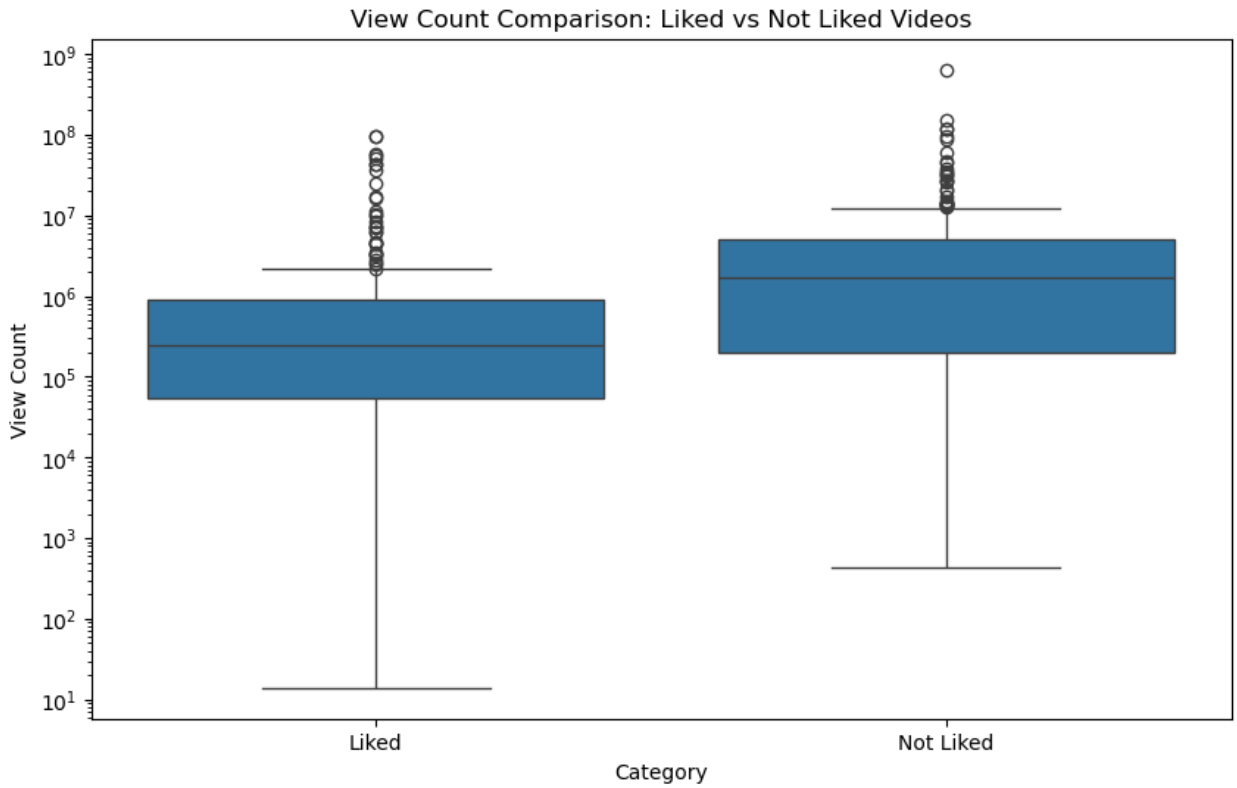
```
Music/history/watch-history.html'

# Get 200 liked videos
liked_videos = get_liked_videos(max_results=200)

# Extract liked video IDs
liked_video_ids = [item['id'] for item in liked_videos]

# Get 1000 recently watched video IDs
recently_watched_video_ids =
get_recently_watched_videos_from_html(file_path, max_results=1000)

# Filter out liked videos
unliked_video_ids = [vid for vid in recently_watched_video_ids if vid
not in liked_video_ids]

# Ensure we have enough videos to sample from
if len(unliked_video_ids) < 200:
    print("Not enough unliked videos to sample from.")
    # Use all available videos
    unliked_sample_ids = unliked_video_ids
else:
    unliked_sample_ids = random.sample(unliked_video_ids, 200)

# Fetch details of the unliked sampled videos
unliked_videos = get_random_videos_details(unliked_sample_ids)

# Create DataFrames for liked and unliked videos
liked_videos_df = create_dataframe(liked_videos, "Liked")
unliked_videos_df = create_dataframe(unliked_videos, "Not Liked")

# Combine both DataFrames
combined_df = pd.concat([liked_videos_df, unliked_videos_df],
ignore_index=True)

# Save DataFrame to CSV
combined_df.to_csv("combined_videos.csv", index=False)

# Display the DataFrame
print(combined_df.head())

# Summarize the dataset
print("\nDataset Summary:")
print(combined_df.describe())

# Bar graph to compare number of liked and unliked videos
plt.figure(figsize=(10, 6))
sns.countplot(x="category", data=combined_df)
plt.title("Number of Videos: Liked vs Not Liked")
plt.xlabel("Category")
```

```
plt.ylabel("Count")
plt.show()

# Compare views of liked and unliked videos
plt.figure(figsize=(10, 6))
sns.boxplot(x="category", y="view_count", data=combined_df)
plt.title("View Count Comparison: Liked vs Not Liked Videos")
plt.xlabel("Category")
plt.ylabel("View Count")
plt.yscale("log")  # Use log scale to handle wide range of view counts
plt.show()
```

```
  category                                                    title  \
0    Liked                                 , 100%
1    Liked      What Is Dynamic Programming and How To Use It
2    Liked  8 vs. 8 soccer: Tactics, Formation, Position (...
3    Liked                                       🐨
4    Liked

                                            description  \
0  #  #  #  #   #  #  \n...
1  **Dynamic Programming Tutorial**\nThis is a qu...
2  This video looks at three of the basics for an...
3
4  # #  #  # # # # \n#Sofa4844...

                                                   tags category_id
duration  \
0  [#  , #  , #  , #  , #  , # ...          26  PT14M11S
1  [dynamic programming tutorial, dynamic program...          27
PT14M28S
2  [soocer, U9, coaching, 3-3-1- formation, kid-f...          22
PT7M20S
3                                                 N/A          22
PT28S
4  [Sofa4844, Shorts,  ,  ,  ,  ,  ,  ...          10     PT29S

   view_count  like_count  comment_count  \
0         132          16              7
1     1606381       41525           1619
2       85312        1167             23
3       58821         986             15
4     1583061       28959           1270

                                     topic_categories language
0          [https://en.wikipedia.org/wiki/Knowledge]       ko
1          [https://en.wikipedia.org/wiki/Knowledge]       en
2  [https://en.wikipedia.org/wiki/Association_foo...      N/A
3               [https://en.wikipedia.org/wiki/Food]      N/A
4  [https://en.wikipedia.org/wiki/Music, https://...       ko
```

```
Dataset Summary:
         view_count    like_count  comment_count
count  3.980000e+02  3.980000e+02     398.000000
mean   7.272980e+06  1.392740e+05    1635.246231
std    3.596421e+07  8.700834e+05   11758.741734
min    1.400000e+01  0.000000e+00       0.000000
25%    9.120475e+04  1.614000e+03      38.000000
50%    5.602590e+05  1.048500e+04     246.000000
75%    3.201264e+06  5.343050e+04     687.000000
max    6.411373e+08  1.484950e+07  197831.000000
```






```
combined_df

      category                                                  title  \
0        Liked                            , 100%
1        Liked      What Is Dynamic Programming and How To Use It
2        Liked  8 vs. 8 soccer: Tactics, Formation, Position (...
3        Liked                                      🐻
4        Liked
..         ...                                                  ...
393  Not Liked
394  Not Liked
395  Not Liked  Crazy? Silicon Valley Software Engineer living...
396  Not Liked                                        (Ghil_Path)
397  Not Liked                                           Potential

                                              description  \
0    #  #  #  #   #  #  \n...
1    **Dynamic Programming Tutorial**\nThis is a qu...
2    This video looks at three of the basics for an...
3
4    #  #  #  # # #  # \n#Sofa4844...
..                                                  ...
393        \n     \n\n□    ...
394  Provided to YouTube by Kwangsoo Media\n\n  ...
395  I have been living in my car for the past 6 mo...
```

```
396  Provided to YouTube by Collab Asia Music\n\n ...
397  Provided to YouTube by The Orchard Enterprises...

                                                     tags category_id
duration  \
0    [#  , #  , #  , #  , #  , # ...          26  PT14M11S
1    [dynamic programming tutorial, dynamic program...          27
PT14M28S
2    [soocer, U9, coaching, 3-3-1- formation, kid-f...          22
PT7M20S
3                                                     N/A          22
PT28S
4    [Sofa4844, Shorts,  ,  ,  ,  ,  ,  ...          10     PT29S
..                                                   ...         ...
...
393  [  ,   , ootd, oodf, ootv,  ,  ,  , ...          24      PT1M
394                       [ (HisWill), ㅅㅏㄹㅇㅏ가ㄷㅏ,    ]          10
PT4M58S
395                                                   N/A          22
PT7M9S
396                      [LEVISTANCE,  ,  (Ghil_Path)]          10
PT4M55S
397  [Lauv, Ari Leff, Michael Pollack, Madison Love...          10
PT2M58S

     view_count  like_count  comment_count  \
0           132          16              7
1       1606381       41525           1619
2         85312        1167             23
3         58821         986             15
4       1583061       28959           1270
..          ...         ...            ...
393     3294142       75574            413
394     1946404        4461              0
395      517509       10414           1425
396      111208         807              5
397     3425649       25632            135

                                       topic_categories language
0            [https://en.wikipedia.org/wiki/Knowledge]       ko
1            [https://en.wikipedia.org/wiki/Knowledge]       en
2    [https://en.wikipedia.org/wiki/Association_foo...      N/A
3                  [https://en.wikipedia.org/wiki/Food]      N/A
4    [https://en.wikipedia.org/wiki/Music, https://...       ko
..                                                  ...      ...
393  [https://en.wikipedia.org/wiki/Entertainment, ...       ko
394  [https://en.wikipedia.org/wiki/Christian_music...      N/A
395  [https://en.wikipedia.org/wiki/Lifestyle_(soci...    en-US
396  [https://en.wikipedia.org/wiki/Christian_music...      N/A
```

```
397  [https://en.wikipedia.org/wiki/Music, https://...      N/A

[398 rows x 11 columns]
```