coffee recommendation:

- our model based on content based filtering technique
- used algorithm are :

  -Label Encoding:This technique converts categorical variables (e.g., "Light", "Medium", "Dark" for roast level) into numerical values so they can be processed by machine learning algorithms.

  -Cosine Similarity:A metric used to measure how similar two vectors are in an n-dimensional space. In this model, it's used to compare the user's preference vector to each coffee option's feature vector, determining which coffee options are most similar to the user's preferences.

  -Truncated Singular Value Decomposition (SVD):An algorithm used for dimensionality reduction. In this model, SVD is applied to reduce the number of features (like roast level, acidity, etc.) while preserving the essential patterns, making it easier to compare the user's preferences with the dataset. This step helps in handling large, sparse matrices more efficiently.

In [ ]:

Importing Libraries: The code imports necessary libraries:

- pandas (as pd) and numpy (as np) for data manipulation.
- LabelEncoder from sklearn.preprocessing for encoding categorical variables.
- cosine_similarity from sklearn.metrics.pairwise to compute similarity between vectors.
- TruncatedSVD from sklearn.decomposition for dimensionality reduction.

- Encoding Categorical Features: A function encode_feature is defined to encode categorical features into numerical values using LabelEncoder. The function fits the encoder to the feature and returns both the encoder and the transformed feature.

Encoding the Dataset:

- An empty dictionary encoders is created to store the encoders for each feature.
- A loop iterates through each feature in the list ['Roast Level', 'Acidity', 'Drink Type', 'Description', 'Drink Time', 'Strength'].
- The features are encoded and stored in new columns with names like Roast Level_Encoded.
- The encoders are saved in the encoders dictionary for later use.

Collecting and Encoding User Preferences:

- The user's preferences are collected using input() prompts.
- Each input is capitalized and encoded using the encode_user_input function. The encoded values are stored in variables like user_roast, user_acidity, etc.

Encoding User Input:

- The function encode_user_input encodes the user's input based on the feature.
- It checks if the input matches one of the recognized classes; if it does, it transforms and returns the encoded value. Otherwise, it prints an error message and returns None.

Validation of User Input:

- The code checks if any user input could not be encoded (i.e., if any variable is None).
- If so, it prints a message asking the user to use the specified labels.

Feature Vector Creation:

- If all inputs are valid, the code creates a feature vector user_vector from the encoded user preferences.
- This vector is converted into a DataFrame user_df.

Compute Similarity:

- The cosine_similarity function calculates the similarity between the user's vector and the encoded features in the DataFrame.

Add Similarity Scores:

- The similarity scores are flattened and added as a new column Similarity to the DataFrame.

Dimensionality Reduction with SVD:

- TruncatedSVD is used to reduce the dimensionality of the feature space to 2 components (n_components=2).
- The encoded features are transformed using SVD, resulting in df_svd for the dat[aset and user_vector_svd for the user's input.

Add SVD-Based Similarity Scores:

- The similarity scores based on the SVD-transformed features are flattened and added as a new column Similarity_SVD in the DataFrame.

Recommend Based on Similarity:

- The DataFrame is sorted by the Similarity_SVD scores in descending order.
- The top 3 recommendations are selected using .head(3).
- The recommended coffee attributes (Flavor, Country, Health Benefit, and Video URL) are printed.

- Error Handling: Added a print statement to notify if user input is not recognized, along with the feature name.
- Input Validation: Included a check to ensure the user input matches the recognized labels.

Troubleshooting

- Check Input: Ensure that the user input exactly matches the expected values. For example, if the options are "Light", "Medium", and "Dark", the input should be one of these exactly, including capitalization.
- Provide Feedback: The code now informs the user when their input is not recognized. This should help users understand what went wrong.

With these updates, the recommendation system should handle user inputs more gracefully and provide better feedback.

In [ ]:

In [ ]:

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.decomposition import TruncatedSVD
```

In [ ]:

In [10]:
```
# # dataset with additional attributes including video URLs, drink time, and strength
# data = {
#     'Flavor': ['Fruity', 'Nutty', 'Chocolatey', 'Earthy', 'Citrus', 'Spicy', 'Floral', 'Herbal', 'Smoky', 'Bold'],
#     'Roast Level': ['Light', 'Medium', 'Dark', 'Medium', 'Light', 'Dark', 'Light', 'Medium', 'Dark', 'Medium'],
#     'Acidity': ['Medium', 'Low', 'High', 'Medium', 'Low', 'High', 'Medium', 'Low', 'High', 'Medium'],
#     'Drink Type': ['Drip', 'Espresso', 'Aeropress', 'Pourover', 'Drip', 'Espresso', 'Aeropress', 'Pourover', 'Drip', 'Espresso'],
#     'Country': ['Ethiopia', 'Colombia', 'Brazil', 'Kenya', 'Costa Rica', 'Honduras', 'Yemen', 'Peru', 'Guatemala', 'Mexico'],
#     'Health Benefit': ['Antioxidant-rich', 'Energy Boost', 'Mood Enhancer', 'Stress Relief', 'Immunity Boost', 'Digestive Aid', 'Anti-inflammatory', 'Focus Improvement', 'Detoxifying', 'Metabolism Boost'],
#     'Description': ['Chocolatey', 'Nutty', 'Fruity', 'Earthy', 'Citrusy', 'Spicy', 'Floral', 'Herbal', 'Smoky', 'Bold'],
#     'Video URL': [
#         'https://example.com/video1',  # Fruity
#         'https://example.com/video2',  # Nutty
#         'https://example.com/video3',  # Chocolatey
#         'https://example.com/video4',  # Earthy
#         'https://example.com/video5',  # Citrus
#         'https://example.com/video6',  # Spicy
#         'https://example.com/video7',  # Floral
#         'https://example.com/video8',  # Herbal
#         'https://example.com/video9',  # Smoky
#         'https://example.com/video10'  # Bold
#     ],
#     'Drink Time': ['Morning', 'Afternoon', 'Evening', 'Morning', 'Afternoon', 'Evening', 'Morning', 'Afternoon', 'Evening', 'Morning'],
#     'Strength': ['Mild', 'Medium', 'Strong', 'Medium', 'Mild', 'Strong', 'Mild', 'Medium', 'Strong', 'Medium']
# }
```

In [3]:
```
# # convert to data frame
# df = pd.DataFrame(data)
# df
```

Out[3]:

	Flavor	Roast Level	Acidity	Drink Type	Country	Health Benefit	Description	Video URL	Drink Time	Strength
0	Fruity	Light	Medium	Drip	Ethiopia	Antioxidant-rich	Chocolatey	https://example.com/video1	Morning	Mild
1	Nutty	Medium	Low	Espresso	Colombia	Energy Boost	Nutty	https://example.com/video2	Afternoon	Medium
2	Chocolatey	Dark	High	Aeropress	Brazil	Mood Enhancer	Fruity	https://example.com/video3	Evening	Strong
3	Earthy	Medium	Medium	Pourover	Kenya	Stress Relief	Earthy	https://example.com/video4	Morning	Medium
4	Citrus	Light	Low	Drip	Costa Rica	Immunity Boost	Citrusy	https://example.com/video5	Afternoon	Mild
5	Spicy	Dark	High	Espresso	Honduras	Digestive Aid	Spicy	https://example.com/video6	Evening	Strong
6	Floral	Light	Medium	Aeropress	Yemen	Anti-inflammatory	Floral	https://example.com/video7	Morning	Mild
7	Herbal	Medium	Low	Pourover	Peru	Focus Improvement	Herbal	https://example.com/video8	Afternoon	Medium
8	Smoky	Dark	High	Drip	Guatemala	Detoxifying	Smoky	https://example.com/video9	Evening	Strong
9	Bold	Medium	Medium	Espresso	Mexico	Metabolism Boost	Bold	https://example.com/video10	Morning	Medium

In [4]:
```
# # Encode categorical features with all possible labels
# def encode_feature(df, feature):
#     encoder = LabelEncoder()
#     encoder.fit(df[feature])
#     return encoder, encoder.transform(df[feature])
```

In [5]:
```
# # Encode the dataset
# encoders = {}
# for feature in ['Roast Level', 'Acidity', 'Drink Type', 'Description', 'Drink Time', 'Strength']:
#     encoder, encoded_feature = encode_feature(df, feature)
#     df[feature + '_Encoded'] = encoded_feature
#     encoders[feature] = encoder
```

In [ ]:

In [6]:
```
# # Collect user preferences
# def encode_user_input(user_input, feature, encoders):
#     encoder = encoders[feature]
#     if user_input in encoder.classes_:
#         return encoder.transform([user_input])[0]
#     else:
#         print(f"Input '{user_input}' is not recognized for {feature}.")
#         return None

# user_roast = encode_user_input(input("Enter your preferred roast level (Light, Medium, Dark): ").capitalize(), 'Roast Level', encoders)
# user_acidity = encode_user_input(input("Enter your preferred acidity level (Low, Medium, High): ").capitalize(), 'Acidity', encoders)
# user_drink = encode_user_input(input("How do you drink your coffee (Drip, Espresso, Aeropress, Pourover): ").capitalize(), 'Drink Type', encoders)
# user_ideal_cup = encode_user_input(input("How would you describe your ideal cup of coffee (Chocolatey, Nutty, Fruity): ").capitalize(), 'Description', encoders)
# user_drink_time = encode_user_input(input("When do you typically drink coffee (Morning, Afternoon, Evening): ").capitalize(), 'Drink Time', encoders)
# user_strength = encode_user_input(input("How strong do you like your coffee (Mild, Medium, Strong): ").capitalize(), 'Strength', encoders)
```

```
Enter your preferred roast level (Light, Medium, Dark): medium
Enter your preferred acidity level (Low, Medium, High): medium
How do you drink your coffee (Drip, Espresso, Aeropress, Pourover): drip
How would you describe your ideal cup of coffee (Chocolatey, Nutty, Fruity): nutty
When do you typically drink coffee (Morning, Afternoon, Evening): afternoon
How strong do you like your coffee (Mild, Medium, Strong): strong
```

In [ ]:

In [9]:
```
# # Check if user input is valid
# if None in [user_roast, user_acidity, user_drink, user_ideal_cup, user_drink_time, user_strength]:
#     print("Some of your inputs are not recognized. Please use the specified labels.")
# else:
#     # Create a feature vector for user preferences
#     user_vector = np.array([user_roast, user_acidity, user_drink, user_ideal_cup, user_drink_time, user_strength])
#     user_df = pd.DataFrame([user_vector], columns=[f + '_Encoded' for f in ['Roast Level', 'Acidity', 'Drink Type', 'Description', 'Drink Time', 'Strength']])

#     # Compute similarity
#     user_similarity = cosine_similarity(user_df, df[[f + '_Encoded' for f in ['Roast Level', 'Acidity', 'Drink Type', 'Description', 'Drink Time', 'Strength']]])

#     # Add similarity scores
#     df['Similarity'] = user_similarity.flatten()

#     # Use SVD for dimensionality reduction
#     svd = TruncatedSVD(n_components=2)
#     df_svd = svd.fit_transform(df[['Roast Level_Encoded', 'Acidity_Encoded', 'Drink Type_Encoded', 'Description_Encoded', 'Drink Time_Encoded', 'Strength_Encoded']])
#     user_vector_svd = svd.transform(user_df)

#     # Compute similarity with SVD
#     user_similarity_svd = cosine_similarity(user_vector_svd, df_svd)
#     df['Similarity_SVD'] = user_similarity_svd.flatten()

#     # Recommend based on SVD similarity
#     top_recommendations = df.sort_values(by='Similarity_SVD', ascending=False).head(3)
#     print("Top Coffee Recommendations:")
#     print(top_recommendations[['Flavor', 'Country', 'Health Benefit', 'Video URL']])
```

```
Top Coffee Recommendations:
   Flavor   Country     Health Benefit                Video URL
1   Nutty  Colombia       Energy Boost  https://example.com/video2
7  Herbal      Peru  Focus Improvement  https://example.com/video8
6  Floral     Yemen  Anti-inflammatory  https://example.com/video7
```

In [ ]:

In [ ]:

In [ ]:

```
# import pandas as pd
# import numpy as np
# from sklearn.preprocessing import LabelEncoder
# from sklearn.metrics.pairwise import cosine_similarity
# from sklearn.decomposition import TruncatedSVD

# # Load the data from a CSV file
# df = pd.read_csv('coffee_data.csv')

# # Encode categorical features with all possible labels
# def encode_feature(df, feature):
#     encoder = LabelEncoder()
#     encoder.fit(df[feature])
#     return encoder, encoder.transform(df[feature])

# # Encode the dataset
# encoders = {}
# for feature in ['Roast Level', 'Acidity', 'Drink Type', 'Description']:
#     encoder, encoded_feature = encode_feature(df, feature)
#     df[feature + '_Encoded'] = encoded_feature
#     encoders[feature] = encoder

# # Collect user preferences
# def encode_user_input(user_input, feature, encoders):
#     encoder = encoders[feature]
#     try:
#         return encoder.transform([user_input])[0]
#     except ValueError:
#         return None

# user_roast = encode_user_input(input("Enter your preferred roast level (Light, Medium, Dark): ").capitalize(), 'Roast Level', encoders)
# user_acidity = encode_user_input(input("Enter your preferred acidity level (Low, Medium, High): ").capitalize(), 'Acidity', encoders)
# user_drink = encode_user_input(input("How do you drink your coffee (Drip, Espresso, Aeropress, Pourover): ").capitalize(), 'Drink Type', encoders)
# user_ideal_cup = encode_user_input(input("How would you describe your ideal cup of coffee (Chocolatey, Nutty, Fruity): ").capitalize(), 'Description', encoders)

# # Check if user input is valid
# if None in [user_roast, user_acidity, user_drink, user_ideal_cup]:
#     print("Some of your inputs are not recognized. Please use the specified labels.")
# else:
#     # Create a feature vector for user preferences
#     user_vector = np.array([user_roast, user_acidity, user_drink, user_ideal_cup])
#     user_df = pd.DataFrame([user_vector], columns=[f + '_Encoded' for f in ['Roast Level', 'Acidity', 'Drink Type', 'Description']])

#     # Compute similarity
#     user_similarity = cosine_similarity(user_df, df[[f + '_Encoded' for f in ['Roast Level', 'Acidity', 'Drink Type', 'Description']]])

#     # Add similarity scores
#     df['Similarity'] = user_similarity.flatten()

#     # Use SVD for dimensionality reduction
#     svd = TruncatedSVD(n_components=2)
#     df_svd = svd.fit_transform(df[['Roast Level_Encoded', 'Acidity_Encoded', 'Drink Type_Encoded', 'Description_Encoded']])
#     user_vector_svd = svd.transform(user_df)

#     # Compute similarity with SVD
#     user_similarity_svd = cosine_similarity(user_vector_svd, df_svd)
#     df['Similarity_SVD'] = user_similarity_svd.flatten()

#     # Recommend based on SVD similarity
#     top_recommendations = df.sort_values(by='Similarity_SVD', ascending=False).head(3)
#     print("Top Coffee Recommendations:")
#     print(top_recommendations[['Flavor', 'Country', 'Health Benefit', 'Video URL']])
```