SO, HOW DO YOU LIKE *YOUR* COFFEE?

University of Western Australia
Data Analytics Bootcamp - 2023

HAVE YOU EVER WONDERED...

What *is* it about your favourite coffee that you enjoy?

Would you know how your preferences might translate into various coffee beans?

If you know you enjoy coffee from Colombia, does that mean you might enjoy coffee from Rwanda?

PROJECT DESCRIPTION

ANALYSTS

Jesslyn Lengkong, Dominique Spencer, Cayley Morrow

SCOPE

This project will create an interactive dashboard where users will select specific criteria using a numeric slider, to define and filter their personal preferences for a coffee bean. The dashboard will then generate the top 5 recommendations tailored to their slider, along with details that can be found on a bag of coffee including the roast level, country of origin, flavour notes, provenance and roaster name.

With the questions posed in the previous slide, the purpose of this project aims for users to become more in tune with their preferences and therefore (hopefully) increase their overall enjoyment of their daily ritual!

The dataset used was created by Tofer Kim, and made available through his *coffee-recommender* repository.
It contains 5,125 rows of data, scraped from a coffee review website, and includes cupping scores on variables such as coffee aroma, flavour, acidity, body and aftertaste.
The reviews date from February 1997 to January 2019.

SOURCES

kaggle

https://www.kaggle.com/datasets/fatihb/coffee-quality-data-cqi

Inspiration for project.

NB - This dataset was not the one used in the final project.

COFFEE REVIEW

https://www.coffeereview.com/

Website where coffee review data was scraped.

Scraping was done and a dataset was created by Tofer Kim, and can be found on GitHub in the repository *coffee-recommender.*

GitHub

https://github.com/toferk/coffee-recommender/tree/master

Repository containing dataset used in this project.

93 Coffee Cycle Roasting

Kenya Gicherori AA

Roaster Location:	San Diego, California
Coffee Origin:	Embu County, Kenya
Roast Level:	Medium-Light
Agtron:	59/74
Est. Price:	$18.50/250 grams

Review Date:	December 2023
Aroma:	9
Acidity/Structure:	8
Body:	8
Flavor:	9
Aftertaste:	9

Blind Assessment

Balanced, sweetly savory. Cocoa nib, bay leaf, goji berry, caramel, pink peppercorn in aroma and cup. Briskly sweet acidity; crisp, satiny mouthfeel. Long, flavor-saturated finish centered around cocoa nib and pink peppercorn notes.

Notes

Produced by members of the Kibugu Farmer's Cooperative Society, from trees of the SL 28, Ruiru 11 and Batian varieties of Arabica. Coffee Cycle Roasting is a San Diego-based specialty roaster founded by cycling enthusiasts. Visit **www.coffeecycleroasting.com** for more information.

DEFINITIONS

01. ‘Cupping’ Score

Rating = cupping score.
This is where coffee professionals test each coffee against the above variables, and an overall score is given to each coffee.

02. Rating (in dataset)

The overall score each coffee achieves out of 100.
≥ 80 is deemed ‘specialty’ coffee.
< 80 is deemed ‘commodity’ coffee.

03. Sliders

User taste preferences based on variables incl. coffee aroma, flavour, acidity, body and aftertaste, out of a score of 10. More info in the recommender app.

04. Desc_1 & Desc_2

Desc_1: Describing the coffee tasting notes.
Desc_2: Describing the coffee provenance.

LIMITATIONS

01. Timeline

Coffee reviews date as far back as 1997, so recommendations (specifically the roaster) may not be completely accurate.

02. Sliders

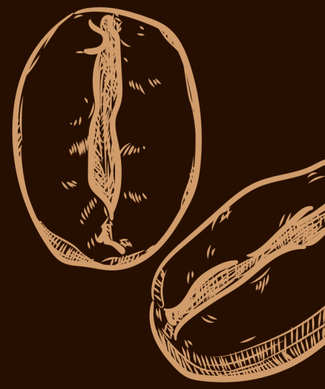
The sliders are fairly vague and could be more specific in terms of preferences!

03. Coffee with milk

Our ratings are for black coffee only, as there weren’t sufficient data points for coffee containing milk.

04. Missing data

Due to null values, we lost 2000+ rows of data. However, the machine learning model won’t have worked if they remained within the file.



CODING APPROACH

FOCUS...

TYPE
The coffee is made from a **single** origin or a **blend** made from multiple origins

ORIGIN/TRACEABILITY
What farm the coffee is from

PROCESSING
Method of removing bean from the fruit

DATE
Date of roast and expiry date

ONE WAY CO2 VALVE
Oxygen will ruin the aroma.

ELEVATION
Altitude at which the beans are grown

ROAST LEVEL
Light/ Medium /Dark (Indicates caffeine level)

EXPECTED TASTE
How the coffee is processed and examples of how the coffee tastes likes.

Initial csv file

[2]:
```
# Read the csv into a Pandas DataFrame
coffee_df = pd.read_csv('../Resources/coffee.csv', encoding='utf-8')
coffee_df.head()
```

[2]:

	all_text	name	rating	roaster	slug	region_africa_arabia	region_caribbean	region_central_america	region_hawaii	region_asia_pacific	...	aroma	acid	body	flavor	aftertaste	with_milk	desc_1	desc_2	desc_3	desc_4
0	\n\n\n\n \n93\nFlight Coffee Co.\nEthiopia Der...	Ethiopia Deri Kochoha	93	Flight Coffee Co.	/review/ethiopia-deri-kochoha-2	1	0	0	0	0	...	9.0	8.0	9.0	9.0	8.0	NaN	Bright, crisp, sweetly tart. Citrus medley, ca...	From the Deri Kochoha mill in the Hagere Marya...	A poised and melodic wet-processed Ethiopia co...	NaN
1	\n\n\n\n\n91\nDoi Chaang Coffee\nEspresso\nLoc...	Espresso	91	Doi Chaang Coffee	/review/espresso-14	0	0	0	0	1	...	8.0	NaN	8.0	8.0	8.0	9.0	Evaluated as espresso. Deeply rich, sweetly ro...	Doi Chaang is a single-estate coffee produced ...	A rich, resonant espresso from Thailand, espec...	NaN
2	\n\n\n\n \n95\nTemple Coffee and Tea\nKenya Ru...	Kenya Ruthaka Peaberry	95	Temple Coffee and Tea	/review/kenya-ruthaka-peaberry	1	0	0	0	0	...	9.0	8.0	9.0	10.0	8.0	NaN	Deeply sweet, richly savory. Dark chocolate, p...	Despite challenges ranging from contested gove...	A high-toned, nuanced Kenya cup, classic in it...	NaN
3	\n\n\n\n \n93\nTemple Coffee and Tea\nEthiopia...	Ethiopia Gora Kone Sidamo	93	Temple Coffee and Tea	/review/ethiopia-gora-kone-sidamo	1	0	0	0	0	...	9.0	8.0	9.0	9.0	8.0	NaN	Fruit-forward, richly chocolaty. Raspberry cou...	Southern Ethiopia coffees like this one are la...	A playful, unrestrained fruit bomb of a coffee...	NaN
4	\n\n\n\n\n93\nChoosy Gourmet\nSpecialty Coffee...	Specialty Coffee Blend Espresso	93	Choosy Gourmet	/review/specialty-coffee-blend-espresso	0	0	0	0	0	...	9.0	NaN	8.0	9.0	8.0	9.0	Evaluated as espresso. Rich, chocolaty, sweetl...	A blend of coffees from Ethiopia (natural-proc...	An espresso blend in which spice notes — in pa...	NaN

5 rows × 34 columns

Columns

We started with 34 columns, and ended with 15 columns.

slug

Unique identifier for each coffee blend.

name

Name of the coffee blend

roaster

Coffee roaster responsible for the blend.

roast

Roast level of the coffee (e.g., light, medium, dark).

country_of_origin

Country where the coffee beans are sourced.

desc_1 and desc_2

Additional information about the coffee's origin.

rating

Rating of the coffee blend.

latitude and longitude

Coffee quality attributes rated by users.

aroma, acid, body, flavor and aftertaste

Coffee quality attributes rated by users.

Before

Work with the 'origin' column:

```
[6]: # Get details about 'origin' column
     coffee_df['origin'].value_counts()
```

```
[6]: Not disclosed.                                    376
     Not disclosed                                    117
     Yirgacheffe growing region, southern Ethiopia.   113
     Boquete growing region, western Panama           109
     Yirgacheffe growing region, southern Ethiopia     94
                                                     ...
     Bururi Province, Burundi.                          1
     La Paz Department, Western Bolivia.                1
     Gakui, Central Kenya.                              1
     Rusizi District, Western Rwanda.                   1
     Indonesia, Central and South America               1
     Name: origin, Length: 1460, dtype: int64
```

After

```
[22]: coffee_df.country_of_origin.value_counts()
```

```
[22]: Ethiopia                                743
      Kenya                                   424
      Indonesia                               370
      Colombia                                334
      Guatemala                               272
      Panama                                  221
      Costa Rica                              190
      Brazil                                  168
      El Salvador                             165
      United States                           165
      Rwanda                                  101
      Nicaragua                                97
      Honduras                                 72
      Papua New Guinea                         63
      Burundi                                  57
      Mexico                                   55
      Peru                                     51
      Thailand                                 49
      Bolivia, Plurinational State of          43
      Tanzania, United Republic of             40
      India                                    37
      Ecuador                                  22
      Jamaica                                  22
      Yemen                                    18
      Congo, The Democratic Republic of the    13
      China                                    12
      Name: country_of_origin, dtype: int64
```

Get Latitude and Longitude coordinates for the dashboard map

[20]:
```
# Create a new DataFrame of the 'country_of_origin' column for getting lat and lon
coffee_countries = coffee_df[['country_of_origin']].copy()

# Extract unique values from the 'Category' column
unique_categories = coffee_countries['country_of_origin'].unique()

# Create a new DataFrame with unique values
unique_df = pd.DataFrame({'country_of_origin': unique_categories})
```

[21]:
```
# Function to get coordinates using pycountry library
def get_coordinates(country):
    try:
        country_obj = pycountry.countries.get(name=country)
        geolocator = Nominatim(user_agent="coffee_countries", timeout=20)
        location = geolocator.geocode(country_obj.name)
        return location.latitude, location.longitude
    except AttributeError:
        return None, None

# Apply function to new Dataframe
unique_df[['latitude', 'longitude']] = unique_df['country_of_origin'].apply(get_coordinates).apply(pd.Series)

# Check the updated DataFrame
unique_df
```

[21]:

	country_of_origin	latitude	longitude
0	Ethiopia	6.767800	35.634371
1	Thailand	13.038762	101.700176
2	Kenya	1.441968	38.431398
3	Honduras	15.257243	-86.075514
4	Congo, The Democratic Republic of the	-2.981434	23.822264
5	Brazil	-10.333333	-53.200000
6	Panama	8.559559	-81.130843
7	Colombia	4.099917	-72.908813
8	Guatemala	15.585555	-90.345759
9	Indonesia	-2.483383	117.890285

Create SQL database

We ended up with 3001 datapoints for our machine learning model

Create SQL database:

[29]:
```
# Define engine path
engine = create_engine('sqlite:///Data_Engineering.db')
```

[30]:
```
# Drop the existing table and create a new one with the desired primary key
with engine.connect() as con:
    con.execute(text('''
        CREATE TABLE IF NOT EXISTS coffee_data (
            "slug" VARCHAR,
            "name" VARCHAR,
            "roaster" VARCHAR,
            "roast" VARCHAR,
            "country_of_origin" VARCHAR,
            "desc_1" VARCHAR,
            "desc_2" VARCHAR,
            "latitude" FLOAT,
            "longitude" FLOAT,
            "rating" INTEGER,
            "aroma" FLOAT,
            "acid" FLOAT,
            "body" FLOAT,
            "flavor" FLOAT,
            "aftertaste" FLOAT,
            "region_africa_arabia" INTEGER,
            "region_caribbean" INTEGER,
            "region_central_america" INTEGER,
            "region_hawaii" INTEGER,
            "region_asia_pacific" INTEGER,
            "region_south_america" INTEGER,
            "type_espresso" INTEGER,
            "type_organic" INTEGER,
            "type_fair_trade" INTEGER,
            "type_decaffeinated" INTEGER,
            "type_pod_capsule" INTEGER,
            "type_blend" INTEGER,
            "type_estate" INTEGER,
            PRIMARY KEY ("slug")
        )
    '''))

# Output to the database
coffee_df.to_sql(name='coffee_data', con=engine, if_exists='replace', index=False)
```

[30]:
```
3001
```

ERD

coffee.csv

Data_Cleaning.ipynb

new_coffee.csv

SQL Database

Flask

Machine Learning

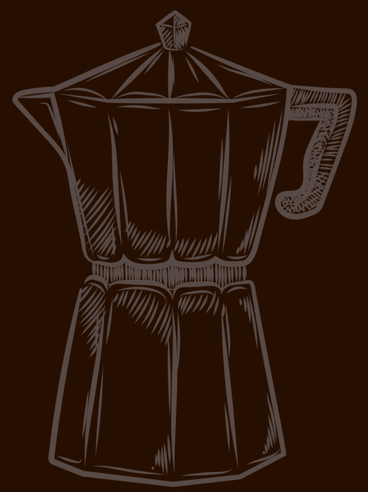
config.toml

streamlit

Dashboard
app.py

scaler.joblib

pca_model.joblib



BUILDING THE RECOMMENDER

Original Setup

aroma

5

0 10

flavor

5

0 10

aftertaste

5

0 10

acidity

5

0 10

body

5

0 10

balance

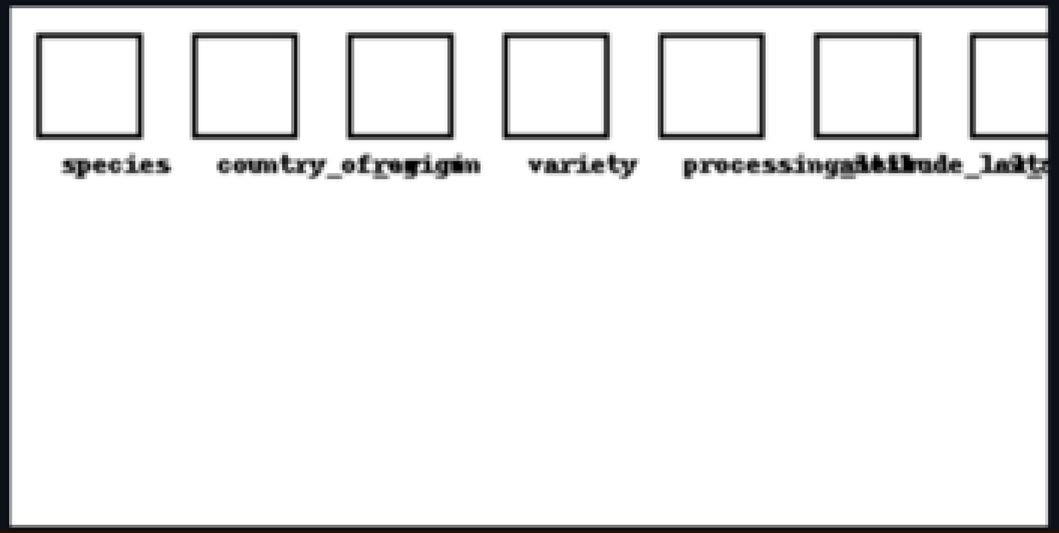
5

0 10

Coffee Preference

This is an example of a coffee recommendation model. This model will generate the top 5 recommendation tailored to the choice of the user. Select the the different coffee components to view your taste.

Custom Image with Sliders



Machine Learning

Specify Features

Standardise Data

PCA application

Save & Incorporate Models

Features

(Rating 0-10)

1. Aroma
2. Acidity
3. Flavour
4. Body
5. Aftertaste

Standardise Data

```
# Create the StandardScaler instance
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

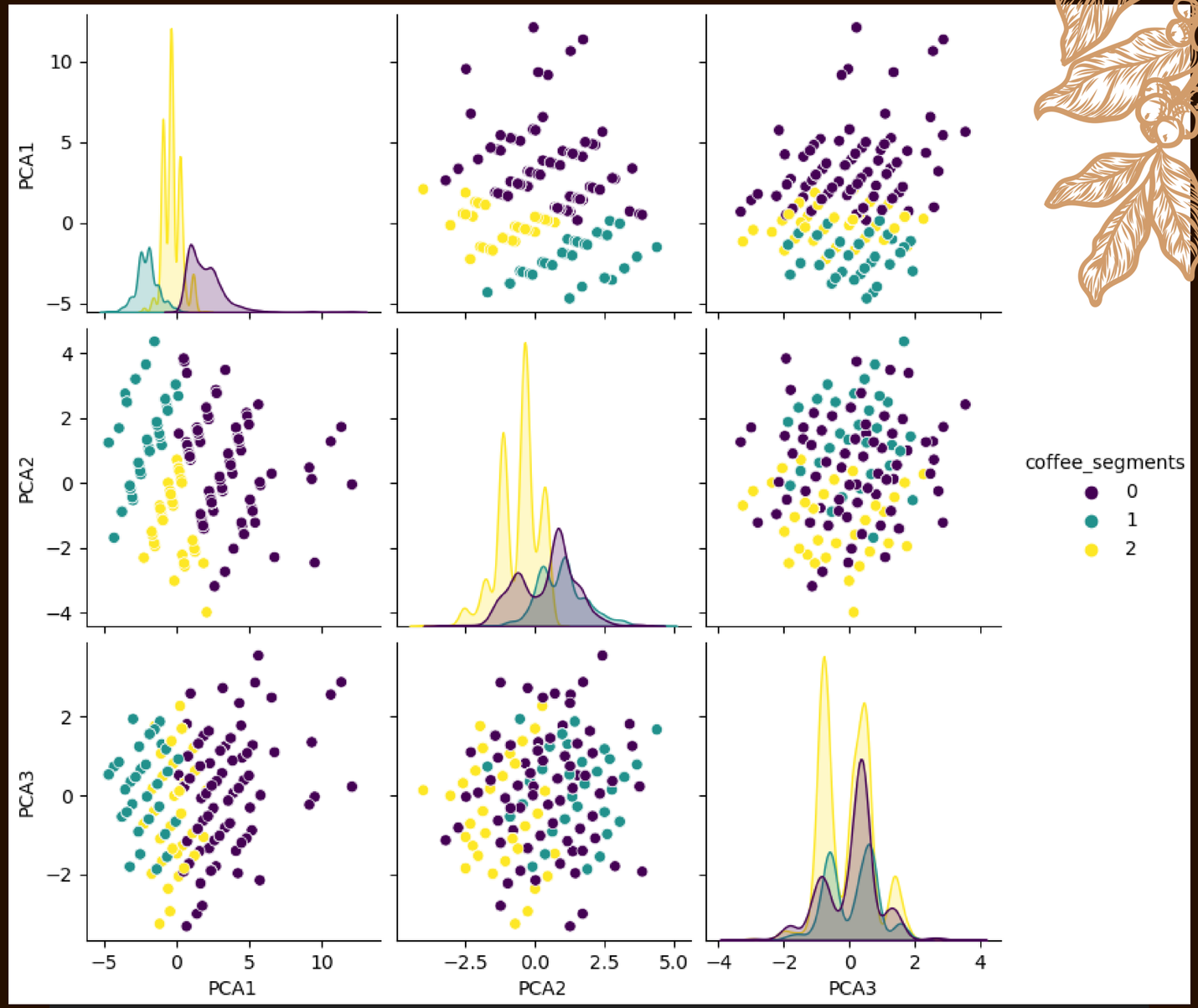
PCA Application

```
# Reduce dimensionality with PCA
pca = PCA(n_components=3)
coffee_data_pca = pca.fit_transform(coffee_data_scaled)

# Calculate the PCA explained variance ratio
explained_variance_ratio = pca.explained_variance_ratio_

# Print the explained variance ratio for each component
print("Explained Variance Ratio for Each Component:")
print(explained_variance_ratio)
```

```
Explained Variance Ratio for Each Component:
[0.57033139 0.22587878 0.12781357]
```

PCA Application

Explained Variance Ratio
= 92.40%



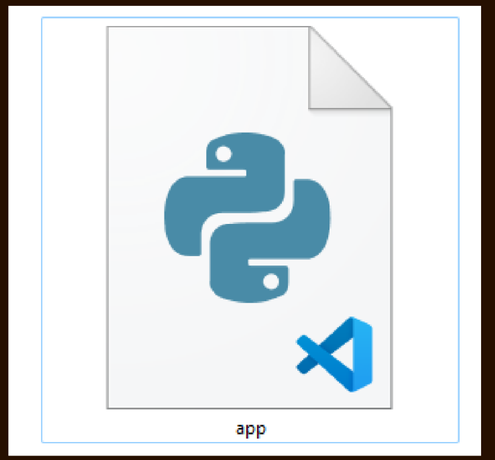

```
Explained Variance Ratio for Each Component:
[0.57033139 0.22587878 0.12781357]
```

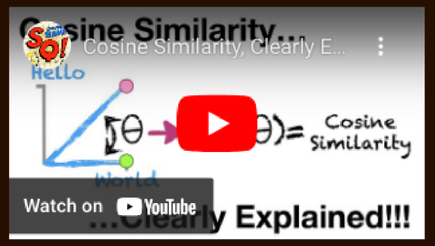
Save & Incorporate Models

.ipynb

```
# Save the models
joblib.dump(model, "models/scaler.joblib")
joblib.dump(coffee_pca_df, "models/pca_model.joblib")
```

Coffee Recommender app.py

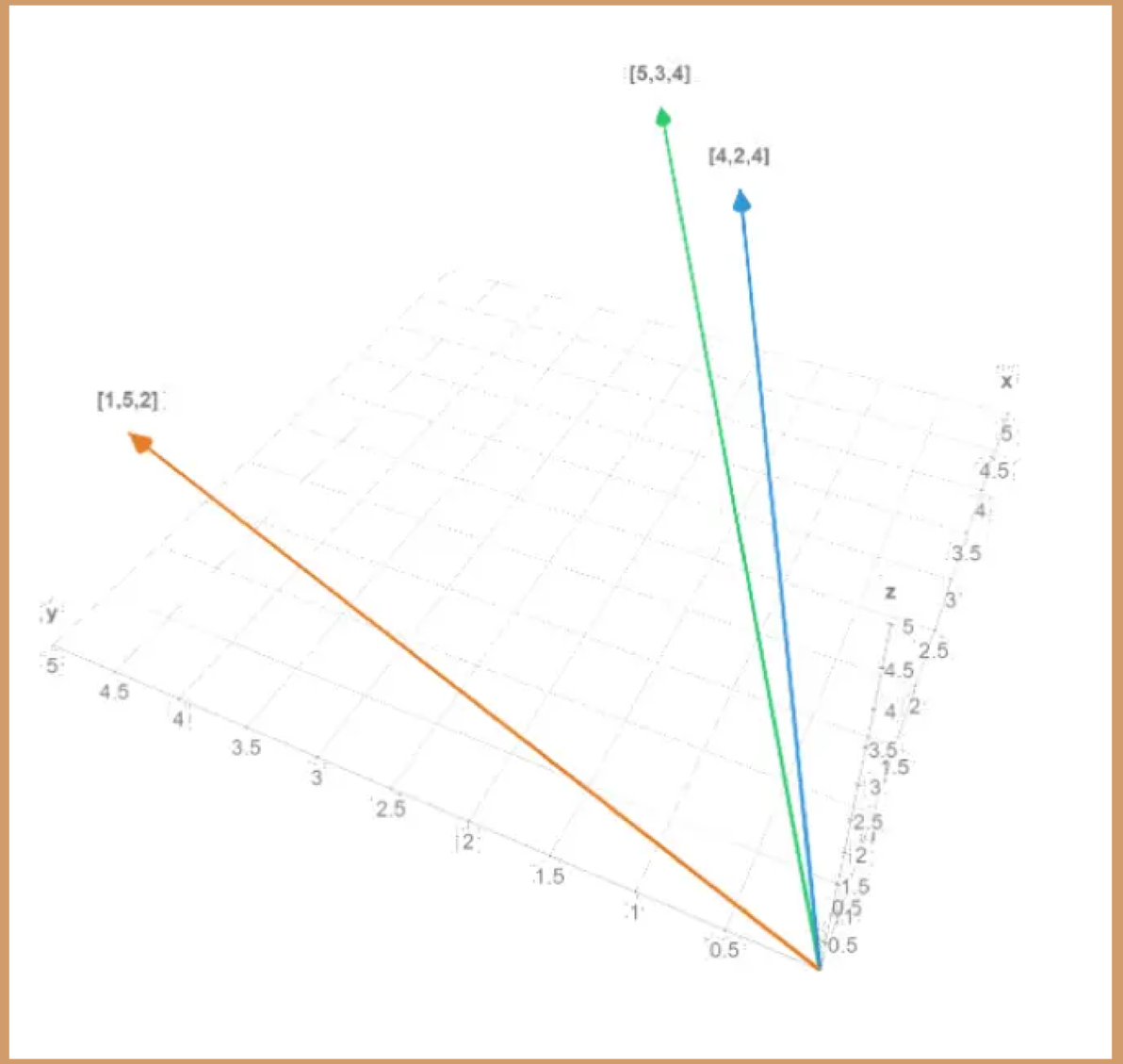


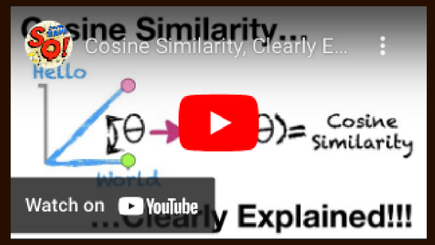


Cosine Similarity

What is it?

Metric that tells us how similar or different things are



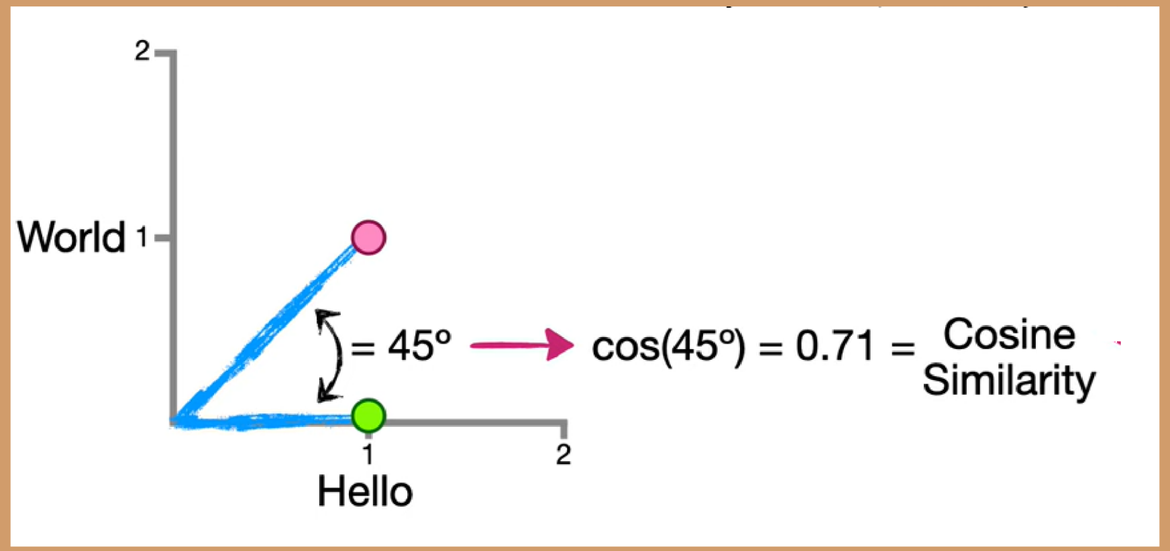


Cosine Similarity

How does it work?

Determines the angle between the two lines plotted on a graph and gives and output between 0 and 1

	Hello	World
Hello, World! →	1	1
Hello! →	1	0



Original Setup

aroma
5
0 10

flavor
5
0 10

aftertaste
5
0 10

acidity
5
0 10

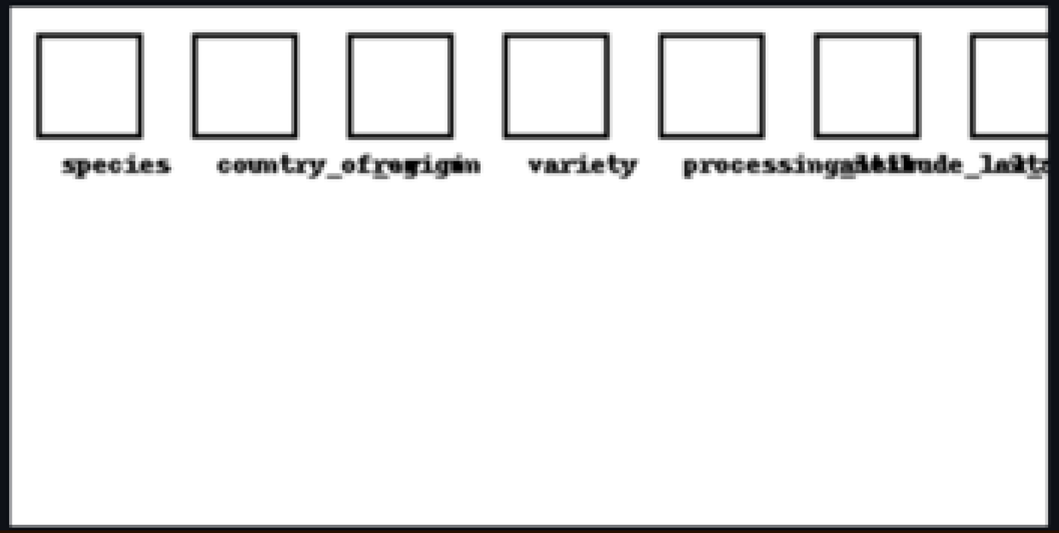
body
5
0 10

balance
5
0 10

Coffee Preference

This is an example of a coffee recommendation model. This model will generate the top 5 recommendation tailored to the choice of the user. Select the the different coffee components to view your taste.

Custom Image with Sliders



What We Developed

Adjust Your Coffee Preferences Here

Select Aroma level
2
0 10

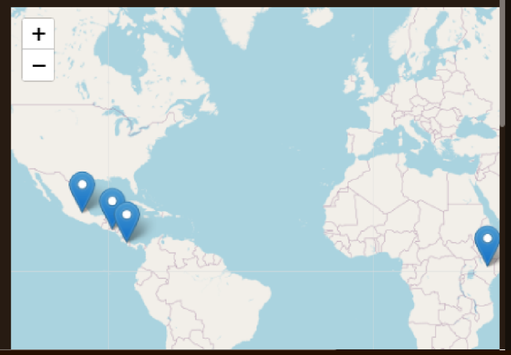
Select Flavour level
1
0 10

Select Acidity level
9
0 10

Select Body level
3
0 10

Select Aftertaste level
4
0 10

Show Me The Coffee! Reset My Results



Coffee Recommender

Welcome to the Coffee Recommender app!

Discover your next favorite coffee based on your preferences.

Adjust the sliders to the left to select your preferred aroma, flavor, acidity, body, and aftertaste levels, and we'll recommend the best coffees for you.

Aroma: The intensity of the coffee's fragrance.
0: No noticeable aroma → 10: Extremely strong and aromatic

Flavour: The overall taste profile of the coffee.
0: No flavour → 10: Intensely rich and complex flavour

Acidity: The perceived brightness or sharpness of the coffee.
0: Low acidity → 10: Very high acidity

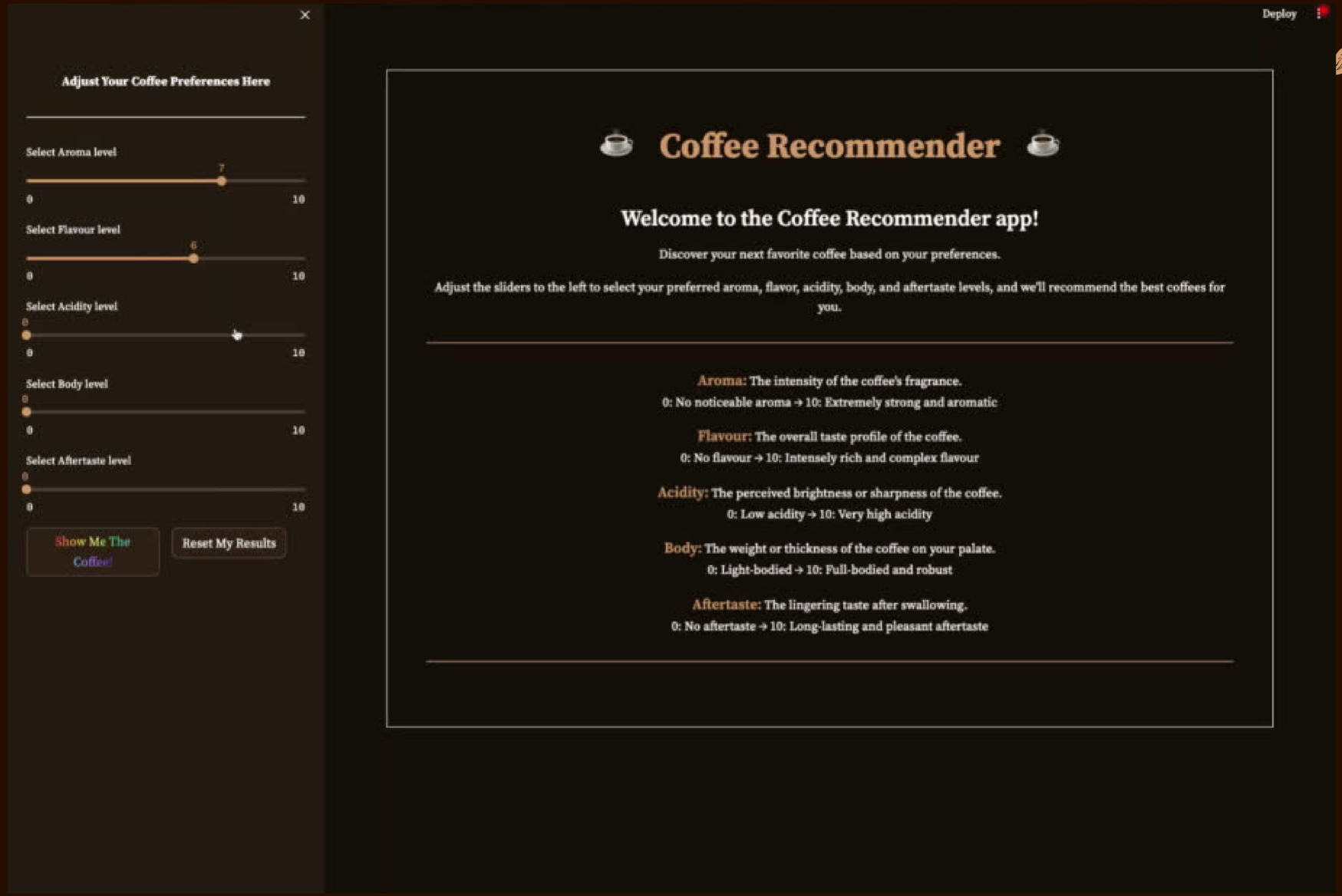
Body: The weight or thickness of the coffee on your palate.
0: Light-bodied → 10: Full-bodied and robust

Aftertaste: The lingering taste after swallowing.
0: No aftertaste → 10: Long-lasting and pleasant aftertaste

Recommendation #1

Name: Mexican Altura Decaf Fair-Trade Organic

Video Recording



SETTING UP A DASHBOARD

```python
# Flask setup
app = Flask(__name__)
CORS(app)

# Database Setup
engine = create_engine("sqlite:///Coding/Data_Engineering.db")

# Reflect the tables
metadata = MetaData()
metadata.reflect(bind=engine)
coffee_data = metadata.tables['coffee_data']

# Fetch data from the database
conn = engine.connect()
result = conn.execute(coffee_data.select()).fetchall()
conn.close()

# Convert each row to a dictionary
data = [dict(row._asdict()) for row in result]

# Create a DataFrame
coffee_data_df = pd.DataFrame(data)
```

```python
# Load Standard Scaler and PCA model
loaded_scaler = joblib.load("models/scaler.joblib")
loaded_pca = joblib.load("models/pca_model.joblib")

# Coffee Recommender
def coffee_recommender(coffee_data_df, loaded_scaler, loaded_pca, aroma, flavor, acid, body, aftertaste, top_n=5):
    input_data = pd.DataFrame({'aroma': [aroma], 'flavour': [flavor], 'acid': [acid], 'body': [body], 'aftertaste': [aftertaste]})  # Apply loaded scaler
    scaled_input_data = loaded_scaler.transform(input_data)  # Apply scaler transformation
    input_pca = loaded_pca.transform(scaled_input_data)  # Apply PCA transformation

    # Transform the entire dataset using PCA
    coffee_data_pca = loaded_pca.transform(coffee_data_df[['aroma', 'flavor', 'acid', 'body', 'aftertaste']])

    # Calculate cosine similarity using the transformed features
    similarity_scores = cosine_similarity(input_pca, coffee_data_pca)

    # Get the indices of the top N similar items
    top_indices = similarity_scores.argsort(axis=1)[:, -top_n:].flatten()

    return coffee_data_df.loc[top_indices]
```

USING STREAMLIT TO CREATE A DASHBOARD

```
# Function to create a Folium map
def create_map(recommended_coffees):
    # Create a map centered at a specific location (you can customize the coordinates)
    map_center = [0, 0]
    coffee_map = folium.Map(location=map_center, zoom_start=2)

    # Add markers for each recommended coffee
    for i, (_, row) in enumerate(recommended_coffees.iterrows(), 1):
        # Customize the content of the popup for each marker
        popup_content = f"<strong>Recommendation #{i}</strong><br><strong>Country of Origin:</strong>
{row['country_of_origin']}<br><strong>Rating:</strong> {row['rating']}"

        marker_location = [row['latitude'], row['longitude']]  # Add latitude and longitude columns to your DataFrame

        folium.Marker(location=marker_location, popup=popup_content).add_to(coffee_map)

    return coffee_map
```

USING STREAMLIT TO CREATE A DASHBOARD

```python
with st.sidebar:
    st.markdown("<div style='text-align: center; font-weight: bold;'>Adjust Your Coffee Preferences Here</div><hr style='border-top: 1px solid #FFFFFF;'>", unsafe_allow_html=True)

    aroma = st.slider(":brown[Select Aroma level]", 0, 10, step=1, key="aroma_slider")
    flavor = st.slider(":brown[Select Flavour level]", 0, 10, step=1, key="flavor_slider")
    acid = st.slider(":brown[Select Acidity level]", 0, 10, step=1, key="acid_slider")
    body = st.slider(":brown[Select Body level]", 0, 10, step=1, key="body_slider")
    aftertaste = st.slider(":brown[Select Aftertaste level]", 0, 10, step=1, key="aftertaste_slider")


    col1, col2 = st.columns(2)  # Create two columns for the buttons to sit in

    # Button 1: Show me my results
    if col1.button(":rainbow[Show Me The Coffee!]"):
        # Get recommended coffees
        recommended_coffees = coffee_recommender(coffee_data_df, loaded_scaler, loaded_pca, aroma, flavor, acid, body, aftertaste, top_n=5)

        # Display recommended coffees on a map
        coffee_map = create_map(recommended_coffees)
        folium_static(coffee_map)

        # Display recommended coffees with centered text
        recommendations_html = ""
        for i, (_, row) in enumerate(recommended_coffees[['rating', 'name', 'roaster', 'roast', 'country_of_origin', 'desc_1', 'desc_2']].head(6).iterrows(), 1):
```

```python
        # Update the content of the container with the recommendations
        result_container.markdown(recommendations_html, unsafe_allow_html=True)

# Button 2: Reset
if col2.button("Reset My Results"):
    # Logic for resetting goes here
    st.warning("Reset")
    # Clear
```

Demonstration

Final Thoughts

- Application of Vectoriation
- Use one-hot encoding to define roasts implementation
- Allow for user review after recommendation given
- Web Scraping application for real-time output
- Show the cosine similarity values in dashboard
- Incorporate model into a mobile app for easier user

```python
def calculate_cosine_similarity(matrix, row_idx1, row_idx2):
    cos_sim = cosine_similarity(matrix[row_idx1, :].reshape(1, -1), matrix[row_idx2, :].reshape(1, -1))[0, 0]
    return cos_sim

# Iterate through rows using iterrows
for idx1, row1 in coffee_data_df.iterrows():
    for idx2, row2 in coffee_data_df.iterrows():
        if idx1 < idx2:  # Avoid comparing the same document and duplicates
            cos_sim = calculate_cosine_similarity(X_desc_1, idx1, idx2)
            print(f'Cosine Similarity between coffee_id {idx1 + 1} and coffee_id {idx2 + 1}: {cos_sim}')
```

```
Cosine Similarity between coffee_id 1 and coffee_id 2: 0.4015578538770222
Cosine Similarity between coffee_id 1 and coffee_id 3: 0.3784378163042982
Cosine Similarity between coffee_id 1 and coffee_id 4: 0.3929197945030012
Cosine Similarity between coffee_id 1 and coffee_id 5: 0.42246305606312096
Cosine Similarity between coffee_id 1 and coffee_id 6: 0.5564148840746572
Cosine Similarity between coffee_id 1 and coffee_id 7: 0.41885390829169555
Cosine Similarity between coffee_id 1 and coffee_id 8: 0.5026246899500346
Cosine Similarity between coffee_id 1 and coffee_id 9: 0.4081649999824365
Cosine Similarity between coffee_id 1 and coffee_id 10: 0.36527478620450904
Cosine Similarity between coffee_id 1 and coffee_id 11: 0.3052601826637909
Cosine Similarity between coffee_id 1 and coffee_id 12: 0.35942537872389224
Cosine Similarity between coffee_id 1 and coffee_id 13: 0.3250056229832959
Cosine Similarity between coffee_id 1 and coffee_id 14: 0.4101124476038661
Cosine Similarity between coffee_id 1 and coffee_id 15: 0.5224292591605755
Cosine Similarity between coffee_id 1 and coffee_id 16: 0.36056136568627406
Cosine Similarity between coffee_id 1 and coffee_id 17: 0.44917077594709154
Cosine Similarity between coffee_id 1 and coffee_id 18: 0.3709432560497715
Cosine Similarity between coffee_id 1 and coffee_id 19: 0.3124666267458028
Cosine Similarity between coffee_id 1 and coffee_id 20: 0.4727085468528941
Cosine Similarity between coffee_id 1 and coffee_id 21: 0.43520348874994297
Cosine Similarity between coffee_id 1 and coffee_id 22: 0.4116109314598473
Cosine Similarity between coffee_id 1 and coffee_id 23: 0.5054405365759586
Cosine Similarity between coffee_id 1 and coffee_id 24: 0.5558097562348634
Cosine Similarity between coffee_id 1 and coffee_id 25: 0.3306232612667903
Cosine Similarity between coffee_id 1 and coffee_id 26: 0.4590703802632966
...
Cosine Similarity between coffee_id 184 and coffee_id 1123: 0.21004201260420144
Cosine Similarity between coffee_id 184 and coffee_id 1124: 0.3717679143145911
Cosine Similarity between coffee_id 184 and coffee_id 1125: 0.49544035707835876
Cosine Similarity between coffee_id 184 and coffee_id 1126: 0.28005601680560194
```

Jay Humor
@jayweingarten

After I drink coffee I show my empty cup to the IT guy and say that I have successfully installed Java. He hates me.




THANK YOU