

The image displays three vertically stacked screenshots of a Jupyter Notebook interface, likely from Google Colab, showing the progression of a data analysis project.

**Screenshot 1:** The first screenshot shows the initial setup and data loading phase. The code cell contains:

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter
import re

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

root_path = '/content/drive/MyDrive/'

Mounted at /content/drive

data_path = 'TM-PROJECT/Uttarakhand_recipes'
data_path

'TM-PROJECT/Uttarakhand_recipes'

df = pd.read_csv(root_path + "TM-PROJECT/uttarakhand_recipes.csv")
df['Ingredients'] = df['Ingredients'].str.lower()
df['Ingredients'] = df['Ingredients'].str.replace('[^a-zA-Z ]', '', regex=True)

df.head()

```

**Screenshot 2:** The second screenshot shows the data frame after cleaning. The code cell contains:

```

df['Ingredient_List'] = df['Ingredients'].apply(lambda x: [i.strip() for i in x.split(',')])

df[['Dish_Name', 'Ingredient_List']].head()

```

The resulting table is:

Dish_Name	Ingredient_List
Kafuli	spinach, fenugreek leaves, mustard oil, garlic...
Bhang Ki Chutney	hemp seeds, yoghurt, lemon juice, garlic, cumi...
Phaanu	mixed pulses, mustard oil, cumin seeds, garlic...
Baadi	urad dal flour, mustard oil, turmeric, cumin s...
Aloo Tamatar Ka Jhol	potatoes, tomatoes, mustard oil, cumin seeds, ...

**Screenshot 3:** The third screenshot shows the implementation of a recommendation system using TF-IDF and cosine similarity. The code cell contains:

```

from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer()
tfidf_matrix = tfidf.fit_transform(df['Ingredients'])

def recommend_similar_dishes(dish_name):
    dish_name = dish_name.lower()
    match = df[df['Dish_Name'].str.lower() == dish_name]

    if match.empty:
        print("Dish not found.")
        return

    idx = match.index[0]
    similarity = cosine_similarity(tfidf_matrix[idx], tfidf_matrix).flatten()
    similar_indices = similarity.argsort()[::-1][1:6]

    print(f"\n{len(similar_indices)} Dishes similar to {dish_name.upper()}:")
    for i in similar_indices:
        print(f"- {df.iat[i]['Dish_Name']}")

from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=4, random_state=42)
df['Cluster'] = kmeans.fit_predict(tfidf_matrix)

df[['Dish_Name', 'Cluster']].head()

```

TMprojectUTTRAKHANDrecipes

```

from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=4, random_state=42)
df['Cluster'] = kmeans.fit_predict(tfidf_matrix)
df[['Dish_Name', 'Cluster']].head()

... Dish_Name Cluster
0 Kafuli 2
1 Bhang Ki Chutney 0
2 Phaanu 0
3 Baadi 0
4 Aaloo Tamatar Ka Jhol 0

all_ingredients = sum(df['Ingredient_List'], [])
ingredient_counts = Counter(all_ingredients)
most_common = ingredient_counts.most_common(10)
most_common

[('mustard oil', 13),
 ('salt', 12),
 ('garlic', 11),
 ('cumin seeds', 10),
 ('turmeric', 9),
 ('jaggery', 5),
 ('green chilli', 3),
 ('rice flour', 2),
 ('yoghurt', 2),
 ('mixed pulses', 2)]

```

Variables Terminal

✓ 1:37 PM Python 3

TMprojectUTTRAKHANDrecipes

```

all_ingredients = sum(df['Ingredient_List'], [])
ingredient_counts = Counter(all_ingredients)
most_common = ingredient_counts.most_common(10)
most_common

[('mustard oil', 13),
 ('salt', 12),
 ('garlic', 11),
 ('cumin seeds', 10),
 ('turmeric', 9),
 ('jaggery', 5),
 ('green chilli', 3),
 ('rice flour', 2),
 ('yoghurt', 2),
 ('mixed pulses', 2)]

ingredients = [item[0] for item in most_common]
counts = [item[1] for item in most_common]

plt.figure(figsize=(10,6))
plt.bar(ingredients, counts)
plt.xticks(rotation=45)
plt.title("Top 10 Most Used Ingredients in Uttarakhand Cuisine")
plt.xlabel("Ingredients")
plt.ylabel("Frequency")
plt.tight_layout()
plt.show()

```

Variables Terminal

✓ 1:37 PM Python 3

TMprojectUTTRAKHANDrecipes

Ingredient	Frequency
mustard oil	13
salt	12
garlic	11
cumin seeds	10
turmeric	9
jaggery	5
green chilli	3
rice flour	2
yoghurt	2
mixed pulses	2

Variables Terminal

✓ 1:37 PM Python 3

```

feature_names = tfidf.get_feature_names_out()
importance = tfidf_matrix.sum(axis=0).A1

ingredient_importance = dict(zip(feature_names, importance))
sorted_importance = sorted(ingredient_importance.items(), key=lambda x: x[1], reverse=True)[:10]

sorted_importance

[('mustard', np.float64(3.107980849248153)),
 ('oil', np.float64(3.063484752875992)),
 ('sesame', np.float64(2.084303896370535)),
 ('ginger', np.float64(2.06589649772551)),
 ('salt', np.float64(2.751749871764717)),
 ('cumin', np.float64(2.706589649772531)),
 ('turmeric', np.float64(2.5745689563582702)),
 ('flour', np.float64(1.7222746516071178)),
 ('jaggery', np.float64(1.675719343902295)),
 ('green', np.float64(1.421687337466999))]

def recommend_dishes(ingredient):
    ingredient = ingredient.lower()
    matching_dishes = df[df['Ingredients'].str.contains(ingredient)]

    if matching_dishes.empty:
        print("No dishes found with this ingredient.")
    else:
        print(f"Dishes containing '{ingredient}': {len(matching_dishes)} results")
        for dish in matching_dishes['Dish_Name']:
            print("-", dish)

    return matching_dishes
  
```

Variables Terminal 1:37 PM Python 3

  

```

user_input = input("Enter an ingredient: ")
matched = recommend_dishes(user_input)

Enter an ingredient: yoghurt
Dishes containing 'yoghurt': (2 results)

- Bhang Ki Chutney
- Gulgula

def show_recipe(dish_name):
    dish = df[df['Dish_Name'].str.lower() == dish_name.lower()]

    if dish.empty:
        print("Dish not found.")
    else:
        print("\nRECIPE FOR:", dish_name.upper())
        print("-" * 50)
        print(dish['Recipe'].values[0])
  
```

Variables Terminal 1:37 PM Python 3

