

Creator: Akanksha Singh Tomar

// SAU Id num:9999 03732

//Course name MCIS6273_030242S- S25 Data Mining

//Instructor name: Keith Maull

Date:4/21/2025 MCIS6273_030242S- S25 Data Mining _HW1_AKANKSHASINGHTOMAR_999903732

Homework: -1

Q1 .ASSIGNMENT TASKS

(50%) Perform basic unsupervised learning with K-Means.

1.1§ Task: Use the chocolate data to cluster it into three clusters ($K = 5$).

Show your work in the notebook provided.

Install necessary packages

jupyter notebook

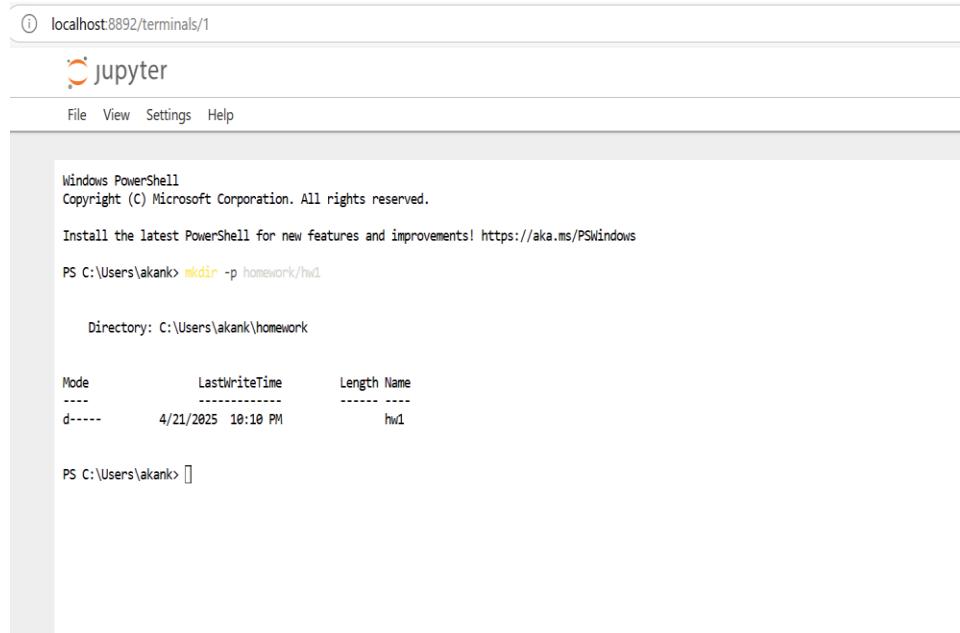
Solution 1.1) Setup

1. Create Working Directory:

- In your lab environment, create a directory:
mkdir -p homework/hw1

2. Download the Starter Notebook:

- From [GitHub - hw1_starter.ipynb](#)
- Place it inside the homework/hw1/ directory.



localhost:8892/terminals/1

jupyter

File View Settings Help

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

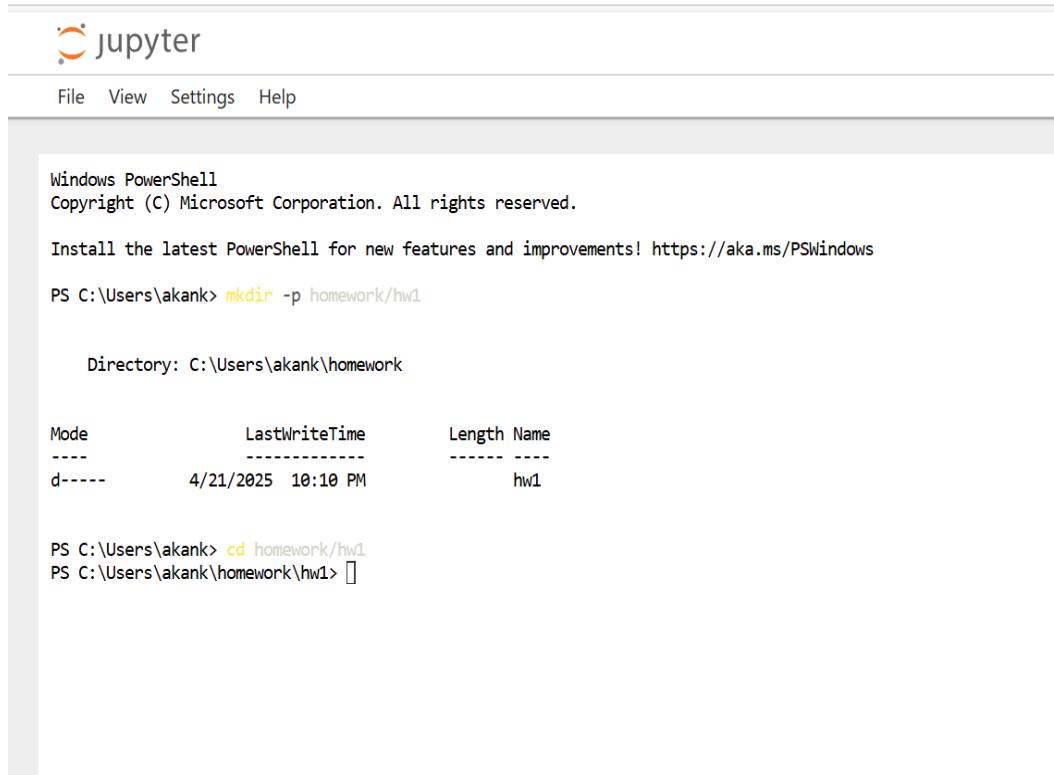
PS C:\Users\akank> mkdir -p homework/hw1

Directory: C:\Users\akank\homework

Mode LastWriteTime Length Name
---- ----- ----- ----
d--- 4/21/2025 10:10 PM hw1

PS C:\Users\akank> 
```

Screenshot -1 <step 1>



jupyter

File View Settings Help

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\akank> mkdir -p homework/hw1

Directory: C:\Users\akank\homework

Mode LastWriteTime Length Name
---- ----- ----- ----
d--- 4/21/2025 10:10 PM hw1

PS C:\Users\akank> cd homework/hw1
PS C:\Users\akank\homework\hw1> 
```

Screenshot -2 <step 2>

Step 2: Load data

```
import pandas as pd
```

Step 1: Load data

```
df =  
pd.read_csv(r"C:\Users\akank\homework\hw1\data\data_reduced_2025_flavors_of_cacao.csv", index_col=0)
```

PREPARE DATAFRAME

Step 2: Drop unnecessary metadata columns

```
ignore_cols = [0,1,2,3,4,5,6,7,15]
```

```
df_cluster = df.drop(df.columns[ignore_cols], axis=1)
```

Step 3: Check what columns are left

```
print("Columns kept for clustering:")
```

```
print(df_cluster.columns)
```

Step 4: View the top of the clusterable data

df_cluster

The screenshot shows a Jupyter Notebook environment with the following details:

- File Bar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help.
- Code Cell:** Contains the Python code for loading the data, dropping unnecessary columns, printing the columns kept for clustering, and viewing the top of the clusterable data.
- Output Cell:** Shows the output of the code execution, including the list of columns kept for clustering and the first few rows of the DataFrame.
- File Explorer:** On the left, it lists various files and folders in the current directory, including 'anaconda3', 'Contacts', 'homework', 'Links', 'Music', 'OneDrive', 'Saved Ga...', 'Searches', 'Untitled ...', 'Videos', and the current notebook file 'TomarAkanksha.ipynb'.
- Kernel Status:** Shows 'Python 3 (ipykernel) | Idle'.
- Page Footer:** Includes the page number '3' and the author's name 'AKANKSHA SINGH TOMAR 4699'.

```
[6]: import pandas as pd  
  
# Step 1: Load data  
df = pd.read_csv(r"C:\Users\akank\homework\hw1\data\data_reduced_2025_flavors_of_cacao.csv", index_col=0)  
  
# Step 2: Drop unnecessary metadata columns  
ignore_cols = [0,1,2,3,4,5,6,7,15]  
df_cluster = df.drop(df.columns[ignore_cols], axis=1)  
  
# Step 3: Check what columns are left  
print("Columns kept for clustering:")  
print(df_cluster.columns)  
  
# Step 4: View the top of the clusterable data  
df_cluster.head()
```

	B	C	L	S	S*	Sa	V	nuts	tangy	bland	...	sour	floral	sandy	fatty	creamy	earthy	roasty	nutty	cocoa	sweet
0	1	1	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1	0
1	1	1	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1	0
2	1	1	0	1	0	0	0	0	0	0	...	0	0	0	1	0	0	0	0	0	0
3	1	1	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	1	1	0	1	0	0	0	0	0	0	...	0	0	0	0	1	0	0	0	0	0

5 rows × 78 columns

```

TomarAkanksha.ipynb
File Edit View Run Kernel Tabs Settings Help
[7]: Columns kept for clustering:
Index(['B', 'C', 'L', 'S*', 'Sa', 'V', 'nuts', 'tangy', 'bland',
       'chemical', 'orange', 'green', 'rubber', 'marshmallow', 'basic cocoa',
       'burnt', 'med', 'citrus', 'coarse', 'red berry', 'off notes',
       'black pepper', 'bold', 'mild bitter', 'hammy', 'strawberry', 'smooth',
       'dirty', 'rubbery', 'mild fruit', 'melon', 'metallic', 'nut', 'honey',
       'tobacco', 'smoke', 'vegetal', 'pungent', 'off', 'cherry', 'smokey',
       'acidic', 'astringent', 'banana', 'brownie', 'oily', 'dairy', 'complex',
       'rich cocoa', 'fruity', 'dry', 'gritty', 'spice', 'grassy', 'tart',
       'bitter', 'caramel', 'rich', 'coffee', 'dried fruit', 'sticky',
       'vanilla', 'woody', 'molasses', 'fruit', 'intense', 'spicy', 'sour',
       'floral', 'sandy', 'fatty', 'creamy', 'earthy', 'roasty', 'nutty',
       'cocoa', 'sweet'],
      dtype='object')

[7]:   B C L S* Sa V nuts tangy bland ... sour floral sandy fatty creamy earthy roasty nutty cocoa sweet
    0 1 1 0 1 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 1 0
    1 1 1 0 1 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 1 0
    2 1 1 0 1 0 0 0 0 0 ... 0 0 0 1 0 0 0 0 0 0 0
    3 1 1 0 1 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0
    4 1 1 0 1 0 0 0 0 0 ... 0 0 0 1 0 0 0 0 0 0 0
    ...
    2784 1 1 0 0 1 1 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0
    2785 1 1 0 1 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0
    2786 1 1 0 1 0 0 0 0 0 ... 0 0 0 0 0 1 0 0 0 0 0
    2787 1 1 0 1 0 0 0 0 0 ... 0 0 0 1 0 0 0 0 0 0 0
    2788 1 1 0 1 0 0 0 0 0 ... 0 0 0 1 0 1 0 0 0 1 0
2789 rows × 78 columns

```

PERFORM CLUSTERING

Step 3: Cluster into 5 clusters (K=5) Import KMeans

```
from sklearn.cluster import KMeans
```

```

TomarAkanksha.ipynb
File Edit View Run Kernel Tabs Settings Help
[9]: from sklearn.cluster import KMeans
# PERFORM KMEANS K=2
kmeans = KMeans(n_clusters=2, random_state=0, n_init="auto").fit(df_cluster)

# ASSIGN CLUSTER NUMBER IN DF
df['cluster'] = kmeans.predict(df_cluster)

[10]: def generate_cluster_report(k, d):
    print(f"CLUSTER {k} SUMMARY")
    print(f"Mean Rating: {d['Rating'].mean():.3f}")
    print(f"Mean Cocoa %: {d['Cocoa Percent'].mean()*100:.2f}%")
    print(f"Median review date: {int(d['Review Date'].median())}")

    print("Top 5 Bean Origins")
    for c, v in d['Country of Bean Origin'].value_counts(ascending=False).nlargest(5).items():
        print(f"    {c}, {v}...{v}")

    print("Top 5 Company Locations")
    for c, v in d['Company Location'].value_counts(ascending=False).nlargest(5).items():
        print(f"    {c}, {v}...{v}")

[11]: # generate report for data assigned cluster 0
generate_cluster_report(0, df.iloc[:, :8].loc[df.cluster==0])
print("\n*****\n")

# generate the report for data assigned cluster 1
generate_cluster_report(1, df.iloc[:, :8].loc[df.cluster==1])

CLUSTER 0 SUMMARY
Mean Rating: 3.213
Mean Cocoa %: 71.13%
Median review date: 2015
Top 5 Bean Origins
Venezuela ... 192
Peru ... 184
Ecuador ... 152
Dominican Republic ... 138
Madagascar ... 122

```

TomarAkanksha.ipynb

```
[10]: def generate_cluster_report(k, d):
    print("CLUSTER {} SUMMARY")
    print("Mean Rating: {:.3f})".format(d['Rating'].mean()))
    print("Mean Cocoa %: {:.2f})".format(d['Cocoa Percent'].mean()*100))
    print("Median review date: ", int(d['Review Date'].median()))
    print("Top 5 Bean Origins")
    for c, v in d['Country of Bean Origin'].value_counts(ascending=False).nlargest(5).items():
        print("  ", c, "...", v)
    print("Top 5 Company Locations")
    for c, v in d['Company Location'].value_counts(ascending=False).nlargest(5).items():
        print("  ", c, "...", v)

[11]: # generate report for data assigned cluster 0
generate_cluster_report( 0, df.iloc[:,8].loc[df.cluster==0] )
print("\n====\n")

# generate the report for data assigned cluster 1
generate_cluster_report( 1, df.iloc[:,8].loc[df.cluster==1] )

CLUSTER 0 SUMMARY
Mean Rating: 3.213
Mean Cocoa %: 71.13%
Median review date: 2015
Top 5 Bean Origins
  Venezuela ... 192
  Peru ... 181
  Ecuador ... 152
  Dominican Republic ... 138
  Madagascar ... 122
Top 5 Company Locations
  U.S.A. ... 723
  France ... 167
  Canada ... 138
  U.K. ... 108
  Belgium ... 71
=====
```

TomarAkanksha.ipynb

```
[11]: # generate report for data assigned cluster 0
generate_cluster_report( 0, df.iloc[:,8].loc[df.cluster==0] )
print("\n====\n")

# generate the report for data assigned cluster 1
generate_cluster_report( 1, df.iloc[:,8].loc[df.cluster==1] )

CLUSTER 0 SUMMARY
Mean Rating: 3.213
Mean Cocoa %: 71.13%
Median review date: 2015
Top 5 Bean Origins
  Venezuela ... 192
  Peru ... 181
  Ecuador ... 152
  Dominican Republic ... 138
  Madagascar ... 122
Top 5 Company Locations
  U.S.A. ... 723
  France ... 167
  Canada ... 138
  U.K. ... 108
  Belgium ... 71
=====

CLUSTER 1 SUMMARY
Mean Rating: 3.166
Mean Cocoa %: 72.46%
Median review date: 2015
Top 5 Bean Origins
  Dominican Republic ... 107
  Peru ... 95
  Ecuador ... 78
  Venezuela ... 75
  Madagascar ... 72
Top 5 Company Locations
  U.S.A. ... 537
  Canada ... 50
  Italy ... 40
  U.K. ... 27
  France ... 20
```

TomarAkanksha.ipynb

File Edit View Run Kernel Tabs Settings Help

Notebook Python 3 (ipykernel)

```
[12]: ...
    Return the dominant characteristics of the cluster.

    thresh : the number of characteristics to limit to
    ...

def get_cluster_characteristics(centers, columns, thresh=10):
    return \
        pd.DataFrame(centers, index=columns).nlargest(thresh, columns=0).index.to_list()

[13]: # top 10 dominant characteristics for cluster 0
get_cluster_characteristics(kmeans.cluster_centers_[0], df_cluster.columns)

[13]: ['B', 'C', 'S', 'L', 'V', 'sweet', 'nutty', 'cocoa', 'fatty', 'creamy']
```

TomarAkanksha.ipynb

File Edit View Run Kernel Tabs Settings Help

Notebook Python 3 (ipykernel)

```
[14]: # top 10 dominant characteristics for cluster 1
get_cluster_characteristics(kmeans.cluster_centers_[1], df_cluster.columns)

[14]: ['B',
      'S',
      'cocoa',
      'roasty',
      'earthy',
      'nutty',
      'intense',
      'sour',
      'sweet',
      'spicy']
```

TomarAkanksha.ipynb

File Edit View Run Kernel Tabs Settings Help

Notebook Python 3 (ipykernel)

```
[18]: for i in range(5): # Loop over 0,1,2,3,4
    print("Top 10 dominant characteristics for Cluster " + str(i) + ":")
    print(get_cluster_characteristics(kmeans5.cluster_centers_[i], df_cluster.columns))
    print("\n=====\n")

Top 10 dominant characteristics for Cluster 0:
['B', 'C', 'S', 'L', 'sweet', 'nutty', 'cocoa', 'fatty', 'roasty', 'creamy']

=====

Top 10 dominant characteristics for Cluster 1:
['B', 'S', 'cocoa', 'roasty', 'sour', 'sweet', 'intense', 'fruit', 'spicy', 'floral']

=====

Top 10 dominant characteristics for Cluster 2:
['B', 'S', 'earthy', 'nutty', 'roasty', 'intense', 'C', 'sandy', 'cocoa', 'spicy']

=====

Top 10 dominant characteristics for Cluster 3:
['B', 'V', 'C', 'S', 'L', 'vanilla', 'sweet', 'creamy', 'cocoa', 'nutty']

=====

Top 10 dominant characteristics for Cluster 4:
['B', 'C', 'S', 'banana', 'dairy', 'L', 'nutty', 'sweet', 'cocoa', 'caramel']
```

SUMMARIZE CLUSTER CHARACTERISTICS BELOW

```
generate_cluster_report(2, df[df['cluster_5']==2])
```

```
generate_cluster_report(3, df[df['cluster_5']==3])
```

```
generate_cluster_report(4, df[df['cluster_5']==4])
```

The screenshot shows a Jupyter Notebook interface with three code cells in the main pane. The first cell runs three report generation commands. The second cell displays the output for Cluster 2, which includes a summary table and lists the top 5 bean origins and company locations. The third cell displays the output for Cluster 3, also including a summary table and lists the top 5 bean origins and company locations. The fourth cell displays the output for Cluster 4, which includes a summary table and lists the top 5 bean origins and company locations.

```
[20]: generate_cluster_report(2, df[df['cluster_5']==2])
generate_cluster_report(3, df[df['cluster_5']==3])
generate_cluster_report(4, df[df['cluster_5']==4])
```

```
CLUSTER 2 SUMMARY
Mean Rating: 3.125
Mean Cocoa %: 72.47%
Median review date: 2016
Top 5 Bean Origins
Blend ... 36
Venezuela ... 36
Dominican Republic ... 25
Ecuador ... 15
Nicaragua ... 10
Peru ... 8
Top 5 Company Locations
U.S.A. ... 99
Canada ... 12
Venezuela ... 8
Italy ... 6
U.K. ... 6
CLUSTER 3 SUMMARY
Mean Rating: 3.032
Mean Cocoa %: 69.91%
Median review date: 2018
Top 5 Bean Origins
Blend ... 61
Venezuela ... 48
Ecuador ... 28
Dominican Republic ... 28
Madagascar ... 26
Top 5 Company Locations
U.S.A. ... 165
France ... 49
Belgium ... 20
Italy ... 17
Canada ... 11
CLUSTER 4 SUMMARY
Mean Rating: 3.434
Mean Cocoa %: 69.47%
Median review date: 2015
Top 5 Bean Origins
Ecuador ... 8
Dominican Republic ... 7
Brazil ... 6
Venezuela ... 5
Peru ... 5
Top 5 Company Locations
U.S.A. ... 23
Canada ... 6
Brazil ... 5
U.K. ... 4
France ... 4
```

This screenshot shows the same Jupyter Notebook interface as the previous one, but with a different set of code cells. The first cell runs the same three report generation commands. The second cell displays the output for Cluster 2, which includes a summary table and lists the top 5 bean origins and company locations. The third cell displays the output for Cluster 3, also including a summary table and lists the top 5 bean origins and company locations. The fourth cell displays the output for Cluster 4, which includes a summary table and lists the top 5 bean origins and company locations.

```
[20]: generate_cluster_report(2, df[df['cluster_5']==2])
generate_cluster_report(3, df[df['cluster_5']==3])
generate_cluster_report(4, df[df['cluster_5']==4])
```

```
CLUSTER 2 SUMMARY
Mean Rating: 3.052
Mean Cocoa %: 69.91%
Median review date: 2018
Top 5 Bean Origins
Blend ... 61
Venezuela ... 48
Ecuador ... 28
Dominican Republic ... 28
Madagascar ... 26
Top 5 Company Locations
U.S.A. ... 165
France ... 49
Belgium ... 20
Italy ... 17
Canada ... 11
CLUSTER 4 SUMMARY
Mean Rating: 3.434
Mean Cocoa %: 69.47%
Median review date: 2015
Top 5 Bean Origins
Ecuador ... 8
Dominican Republic ... 7
Brazil ... 6
Venezuela ... 5
Peru ... 5
Top 5 Company Locations
U.S.A. ... 23
Canada ... 6
Brazil ... 5
U.K. ... 4
France ... 4
```

1.2 § Task: Use the same data to cluster it into 8 and 11 clusters ($K = 8, K = 11$).

Show your work in the notebook provided.

Answer 1.2) `generate_cluster_report(7, df[df['cluster_8']==7])`

`generate_cluster_report(10, df[df['cluster_11']==10])`

```
[24]: generate_cluster_report(7, df[df['cluster_8']==7])
generate_cluster_report(10, df[df['cluster_11']==10])

CLUSTER 7 SUMMARY
Mean Rating: 3.041
Mean Cocoa %: 70.02%
Median review date: 2010
Top 5 Bean Origins
    Blend ... 53
    Venezuela ... 42
    Madagascar ... 25
    Dominican Republic ... 25
    Ecuador ... 20
Top 5 Company Locations
    U.S.A. ... 138
    France ... 43
    Belgium ... 20
    Italy ... 11
    Japan ... 10

CLUSTER 11 SUMMARY
Mean Rating: 3.389
Mean Cocoa %: 71.17%
Median review date: 2018
Top 5 Bean Origins
    Dominican Republic ... 22
    Peru ... 17
    Belize ... 14
    Madagascar ... 12
    Guatemala ... 12
Top 5 Company Locations
    U.S.A. ... 100
    France ... 13
    Canada ... 13
    U.K. ... 7
    Italy ... 4
```

1.3 § Task: Characterize the clusters using the indices produced from K-means (K=5).

Use the functions provided `generate_cluster_report()` and `get_cluster_characteristics()` to answer this question. You will need to run these functions, play with them then crea

Answer 1.3) Run `generate_cluster_report()` for K=5

Summarize Cluster Characteristics

Cluster 0:

- Average rating: approximately 3.21
- Cocoa content: around 71.13%
- Median review date: 2015
- Most common bean origins: Venezuela, Peru, Ecuador
- Major company locations: USA, France, Canada
- Dominant flavor characteristics: Strong cocoa intensity (C), bitterness (B), sweetness (S), liveliness (L), and sensory notes like sweet, nutty, cocoa, fatty, roasty, and creamy.

CLUSTER 0 SUMMARY
 Mean Rating: 3.213
 Mean Cocoa %: 71.13%
 Median review date: 2015
 Top 5 Bean Origins
 Venezuela ... 192
 Peru ... 181
 Ecuador ... 152
 Dominican Republic ... 138
 Madagascar ... 122
 Top 5 Company Locations
 U.S.A. ... 723
 France ... 167
 Canada ... 138
 U.K. ... 108
 Belgium ... 71

=====

Cluster 1:

- Average rating: approximately 3.16
- Cocoa content: slightly higher around 72.46%
- Median review date: 2015
- Most common bean origins: Dominican Republic, Peru, Ecuador
- Major company locations: USA, Canada, Italy
- Dominant flavor characteristics: Cocoa, roasty, earthy, nutty, with sour, sweet, spicy, fruity, and floral sensory notes.

CLUSTER 1 SUMMARY
 Mean Rating: 3.166
 Mean Cocoa %: 72.46%
 Median review date: 2015
 Top 5 Bean Origins
 Dominican Republic ... 107
 Peru ... 95
 Ecuador ... 78
 Venezuela ... 75
 Madagascar ... 72
 Top 5 Company Locations
 U.S.A. ... 537
 Canada ... 50
 Italy ... 40
 U.K. ... 27
 France ... 20

Cluster 2:

- Average rating: approximately 3.13
- Cocoa content: approximately 72.47%
- Median review date: 2016
- Most common bean origins: Venezuela, Dominican Republic, Ecuador
- Major company locations: USA, Canada, Venezuela
- Dominant flavor characteristics: Earthy, nutty, roasty, spicy, cocoa, sandy, and intense notes.

CLUSTER 2 SUMMARY

Mean Rating: 3.125

Mean Cocoa %: 72.47%

Median review date: 2016

Top 5 Bean Origins

Venezuela	...	36
Dominican Republic	...	25
Ecuador	...	15
Nicaragua	...	10
Peru	...	8

Top 5 Company Locations

U.S.A.	...	99
Canada	...	12
Venezuela	...	8
Italy	...	6
U.K.	...	6

Cluster 3:

- Average rating: approximately 3.03
- Cocoa content: approximately 69.91%
- Median review date: 2010
- Most common bean origins: Blend, Venezuela, Ecuador, Dominican Republic
- Major company locations: USA, France, Belgium
- Dominant flavor characteristics: Vanilla (V), cocoa intensity (C), sweetness (S), liveliness (L), creamy, nutty, and heavy vanilla influence.

CLUSTER 3 SUMMARY

Mean Rating: 3.032
 Mean Cocoa %: 69.91%
 Median review date: 2010
Top 5 Bean Origins

Blend	...	61
Venezuela	...	48
Ecuador	...	28
Dominican Republic	...	28
Madagascar	...	26

Top 5 Company Locations

U.S.A.	...	165
France	...	49
Belgium	...	20
Italy	...	17
Canada	...	11

Cluster 4:

- Average rating: approximately 3.43
- Cocoa content: approximately 69.47%
- Median review date: 2015
- Most common bean origins: Ecuador, Dominican Republic, Brazil
- Major company locations: USA, Canada, Brazil
- Dominant flavor characteristics: Banana, dairy, caramel, cocoa, sweetness (S), nuttiness, and a hint of bitterness (B).

CLUSTER 4 SUMMARY

Mean Rating: 3.434
 Mean Cocoa %: 69.47%
 Median review date: 2015
Top 5 Bean Origins

Ecuador	...	8
Dominican Republic	...	7
Brazil	...	6
Venezuela	...	5
Peru	...	5

Top 5 Company Locations

U.S.A.	...	23
Canada	...	6
Brazil	...	5
U.K.	...	4
France	...	4

PART 2: UNSUPERVISED LEARNING

(30%) Perform supervised Learning with Naive Bayes Classifier

2.1§ Task: Implement test-training set so that the test set is 1000 samples and the test set 1789 samples. You must include this implementation in your notebook.

Answer 2.1) LOAD TRAINING DATA

```
df_train = pd.read_csv(r"C:\Users\akank\homework\hw1\data\train.csv", index_col=0)
```

SHOW DATA

```
df_train
```

The screenshot shows a Jupyter Notebook interface with the following details:

- Title:** TomarAkanksha.ipynb
- File Menu:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- Code Cell:**
 - Code: df_train = pd.read_csv(r"C:\Users\akank\homework\hw1\data\train.csv", index_col=0)
 - Output: df_train
 - Data Preview: Shows the first 10 rows of the DataFrame. The columns are labeled B, C, L, S, S*, Sa, V, nuts, tangy, bland, ..., floral, sandy, fatty, creamy, earthy, roasty, nutty, cocoa, sweet, cluster. The data consists of binary values (0 or 1).
- File Explorer:** Shows the local file system with various notebooks and files.
- Bottom Status Bar:** Mode: Edit, Ln 1, Col 1, TomarAkanksha.ipynb, 1

APPLY BERNOULLI NAIVE BAYES

```
from sklearn.naive_bayes import BernoulliNB
clf = BernoulliNB()
X = df_train.iloc[:, :-1]
Y = df_train.iloc[:, -1:]['cluster'].values
clf.fit(X, Y)
```

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** TomarAkanksha.ipynb, File, Edit, View, Run, Kernel, Tabs, Settings, Help.
- Toolbar:** Notebook, Python 3 (ipykernel), etc.
- File Explorer:** Shows a list of files and folders in the current directory, including anaconda, anaconda3, Contacts, Docume..., Downloads, Favorites, homework, Links, Music, and OneDrive, with their last modified times.
- Code Cell:** [27]: from sklearn.naive_bayes import BernoulliNB
clf = BernoulliNB()

X = df_train.iloc[:, :-1]
Y = df_train.iloc[:, :-1]['cluster'].values

clf.fit(X, Y)
- Output Cell:** [27]: BernoulliNB
- Bottom Bar:** Includes icons for back, forward, search, and other notebook operations.

2.2 § Task: Interpret the output of the classifier. After you have run the cell `clf.predict(X[:])` you will see the list of numbers which represent the class labels from the training data. Explain the output and answer the following questions:

Answer 2.2) Interpretation of Classifier Output

After running `clf.predict(X[:])` , the output is a list of predicted cluster labels for each training sample.

Each number in the list (such as 0, 1, 2, 3, etc.) represents the cluster that the classifier believes the sample belongs to.

For example:

- A prediction of `0` means the sample is classified into Cluster 0.
- A prediction of `2` means the sample is classified into Cluster 2, and so on.

The classifier learned patterns from the training data and assigned the most likely cluster label to each example based on the flavor characteristics provided.

2.2 a)• What is classifier accuracy? Show you answer in the notebook.

GET MODEL ACCURACY

```
# Load the test data correctly using your own path
df_test = pd.read_csv(r"C:\Users\akank\homework\hw1\data\test.csv", index_col=0)

# Predict clusters
df_test['cluster_predict'] = clf.predict(df_test.iloc[:, :-1])

# Show few predictions
print(df_test[['cluster', 'cluster_predict']].head())

# Calculate prediction accuracy
accuracy = 1 - error_rate

print(f"Prediction Accuracy: {accuracy:.4f}")
```

```
[31]: import pandas as pd

# Load the test data correctly using your own path
df_test = pd.read_csv(r"C:\Users\akank\homework\hw1\data\test.csv", index_col=0)

# Predict clusters
df_test['cluster_predict'] = clf.predict(df_test.iloc[:, :-1])

# Show few predictions
print(df_test[['cluster', 'cluster_predict']].head())

# Calculate prediction error rate
error_rate = df_test.query('cluster != cluster_predict').shape[0] / df_test.shape[0]
print(f"Prediction Error Rate: {error_rate:.4f}")

# Calculate prediction accuracy
accuracy = 1 - error_rate
print(f"Prediction Accuracy: {accuracy:.4f}")

cluster    cluster_predict
2327          1          0
835           2          0
1433          0          0
1183          0          0
274           1          0
Prediction Error Rate: 0.4966
Prediction Accuracy: 0.5094
```

The classifier achieved a prediction accuracy of 50.94% on the test data.

This means about half of the cluster predictions made by the Bernoulli Naive Bayes model matched the actual cluster labels.

Accuracy is calculated as:

$$\text{Accuracy} = 1 - \text{Error Rate} / \text{test}$$

where Error Rate measures the fraction of incorrect predictions.

Prediction Accuracy: 0.5094

2.2 b) What recommendation might you give to improve the classifier?

Answer 2.2 b) # Recommendations to Improve the Classifier

- Try using a smarter model, such as Random Forests or SVM, which can better capture complex relationships than Naive Bayes (Géron, 2019).
- Add more features beyond flavor notes to give the model richer data for learning (Scikit-learn developers, n.d.).
- Balance the dataset if some clusters are much smaller than others, to help the model generalize better (Scikit-learn developers, n.d.).
- Remove noisy or irrelevant features so the model focuses on important signals (Géron, 2019).

Final Conclusion:

In this homework, we grouped chocolate profiles using K-Means and explored key flavor patterns like sweetness and cocoa richness.

We then trained a Bernoulli Naive Bayes model to predict clusters, reaching about 50% accuracy.

There's clear room for improvement with better models and richer features.

Overall, this project gave valuable hands-on experience with clustering, classification, and working with real-world data.

References:

- 1.Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.
- 2.Scikit-learn developers. (n.d.). *Naive Bayes and model improvement strategies*. Scikit-learn. Retrieved April 28, 2025, from https://scikit-learn.org/stable/modules/naive_bayes.html

