# Coffee Quality Data (CQI May-2023)

## 資料集 ¶

In [2]:

```
!gdown --id 1-Snv-zd_aCoFLv2n67hjz42MP77bLHNI
```

```
/usr/local/lib/python3.10/dist-packages/gdown/cli.py:121: FutureWar
ning: Option `--id` was deprecated in version 4.3.1 and will be rem
oved in 5.0. You don't need to pass it anymore to use a file ID.
  warnings.warn(
Downloading...
From: https://drive.google.com/uc?id=1-Snv-zd_aCoFLv2n67hjz42MP77bL
HNI (https://drive.google.com/uc?id=1-Snv-zd_aCoFLv2n67hjz42MP77bLH
NI)
To: /content/df_arabica_clean.csv
100% 113k/113k [00:00<00:00, 113MB/s]
```

In [62]:

```
import pandas as pd
import numpy as np
```

In [4]:

```
col_use = ['Country of Origin', 'Variety', 'Processing Method', 'Color',
       'Aroma', 'Flavor', 'Aftertaste', 'Acidity', 'Body', 'Balance',
       'Uniformity', 'Clean Cup', 'Sweetness', 'Overall', 'Defects',
       'Total Cup Points', 'Moisture Percentage', 'Category One Defects',
       'Quakers', 'Category Two Defects']

df_coffee = pd.read_csv('df_arabica_clean.csv', usecols=col_use)
df_coffee
```

Out[4]:

| | Country of Origin | Variety | Processing Method | Aroma | Flavor | Aftertaste | Acidity | Body |
|---|---|---|---|---|---|---|---|---|
| **0** | Colombia | Castillo | Double Anaerobic Washed | 8.58 | 8.50 | 8.42 | 8.58 | 8.25 |
| **1** | Taiwan | Gesha | Washed / Wet | 8.50 | 8.50 | 7.92 | 8.00 | 7.92 |
| **2** | Laos | Java | Semi Washed | 8.33 | 8.42 | 8.08 | 8.17 | 7.92 |
| **3** | Costa Rica | Gesha | Washed / Wet | 8.08 | 8.17 | 8.17 | 8.25 | 8.17 |
| **4** | Colombia | Red Bourbon | Honey,Mossto | 8.33 | 8.33 | 8.08 | 8.25 | 7.92 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **202** | Brazil | Mundo Novo | Natural / Dry | 7.17 | 7.17 | 6.92 | 7.17 | 7.42 |
| **203** | Nicaragua | SHG | Natural / Dry | 7.33 | 7.08 | 6.75 | 7.17 | 7.42 |
| **204** | Laos | Catimor | Washed / Wet | 7.25 | 7.17 | 7.08 | 7.00 | 7.08 |
| **205** | El Salvador | Maragogype | Natural / Dry | 6.50 | 6.75 | 6.75 | 7.17 | 7.08 |
| **206** | Brazil | Mundo Novo | SEMI-LAVADO | 7.25 | 7.08 | 6.67 | 6.83 | 6.83 |

207 rows × 20 columns

變數多時可以先拉第一行出來看一下資料

In [5]:

```
df_coffee.iloc[0]
```

Out[5]:

```
Country of Origin                      Colombia
Variety                                Castillo
Processing Method       Double Anaerobic Washed
Aroma                                      8.58
Flavor                                      8.5
Aftertaste                                 8.42
Acidity                                    8.58
Body                                       8.25
Balance                                    8.42
Uniformity                                 10.0
Clean Cup                                  10.0
Sweetness                                  10.0
Overall                                    8.58
Defects                                     0.0
Total Cup Points                          89.33
Moisture Percentage                        11.8
Category One Defects                          0
Quakers                                       0
Color                                     green
Category Two Defects                          3
Name: 0, dtype: object
```

In [6]:

```
df_coffee.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 207 entries, 0 to 206
Data columns (total 20 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Country of Origin      207 non-null    object 
 1   Variety                201 non-null    object 
 2   Processing Method      202 non-null    object 
 3   Aroma                  207 non-null    float64
 4   Flavor                 207 non-null    float64
 5   Aftertaste             207 non-null    float64
 6   Acidity                207 non-null    float64
 7   Body                   207 non-null    float64
 8   Balance                207 non-null    float64
 9   Uniformity             207 non-null    float64
 10  Clean Cup              207 non-null    float64
 11  Sweetness              207 non-null    float64
 12  Overall                207 non-null    float64
 13  Defects                207 non-null    float64
 14  Total Cup Points       207 non-null    float64
 15  Moisture Percentage    207 non-null    float64
 16  Category One Defects   207 non-null    int64  
 17  Quakers                207 non-null    int64  
 18  Color                  207 non-null    object 
 19  Category Two Defects   207 non-null    int64  
dtypes: float64(13), int64(3), object(4)
memory usage: 32.5+ KB
```

In [7]:

```
df_coffee.describe()
```

Out[7]:

| | Aroma | Flavor | Aftertaste | Acidity | Body | Balance | Unif |
|---|---|---|---|---|---|---|---|
| **count** | 207.000000 | 207.000000 | 207.000000 | 207.00000 | 207.000000 | 207.000000 | 207.0 |
| **mean** | 7.721063 | 7.744734 | 7.599758 | 7.69029 | 7.640918 | 7.644058 | 9.9 |
| **std** | 0.287626 | 0.279613 | 0.275911 | 0.25951 | 0.233499 | 0.256299 | 0.1 |
| **min** | 6.500000 | 6.750000 | 6.670000 | 6.83000 | 6.830000 | 6.670000 | 8.6 |
| **25%** | 7.580000 | 7.580000 | 7.420000 | 7.50000 | 7.500000 | 7.500000 | 10.0 |
| **50%** | 7.670000 | 7.750000 | 7.580000 | 7.67000 | 7.670000 | 7.670000 | 10.0 |
| **75%** | 7.920000 | 7.920000 | 7.750000 | 7.87500 | 7.750000 | 7.790000 | 10.0 |
| **max** | 8.580000 | 8.500000 | 8.420000 | 8.58000 | 8.250000 | 8.420000 | 10.0 |

# 清整資料

由上面的基本統計值可以知道 Defects 的值都是 0，而 Clean Cup 和 Sweetness 全部值都是 10。所以這邊先移除這幾個欄位

In [8]:

```
df_coffee.drop(['Defects', 'Clean Cup', 'Sweetness'], axis=1, inplace=True)
```

其中只有品種(Variety) 的部份有一些空值。這邊直接用數量最多的補值

In [9]:

```
from sklearn.impute import SimpleImputer

cat_imputer = SimpleImputer(strategy="most_frequent")

cat_list = df_coffee.select_dtypes(include=["object"]).columns.tolist() # 選擇類別
print('Categorical features:', cat_list)
df_coffee[cat_list] = cat_imputer.fit_transform(df_coffee[cat_list])
```

```
Categorical features: ['Country of Origin', 'Variety', 'Processing
Method', 'Color']
```

In [10]:

```
df_coffee.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 207 entries, 0 to 206
Data columns (total 17 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Country of Origin      207 non-null    object 
 1   Variety                207 non-null    object 
 2   Processing Method      207 non-null    object 
 3   Aroma                  207 non-null    float64
 4   Flavor                 207 non-null    float64
 5   Aftertaste             207 non-null    float64
 6   Acidity                207 non-null    float64
 7   Body                   207 non-null    float64
 8   Balance                207 non-null    float64
 9   Uniformity             207 non-null    float64
 10  Overall                207 non-null    float64
 11  Total Cup Points       207 non-null    float64
 12  Moisture Percentage    207 non-null    float64
 13  Category One Defects   207 non-null    int64  
 14  Quakers                207 non-null    int64  
 15  Color                  207 non-null    object 
 16  Category Two Defects   207 non-null    int64  
dtypes: float64(10), int64(3), object(4)
memory usage: 27.6+ KB
```

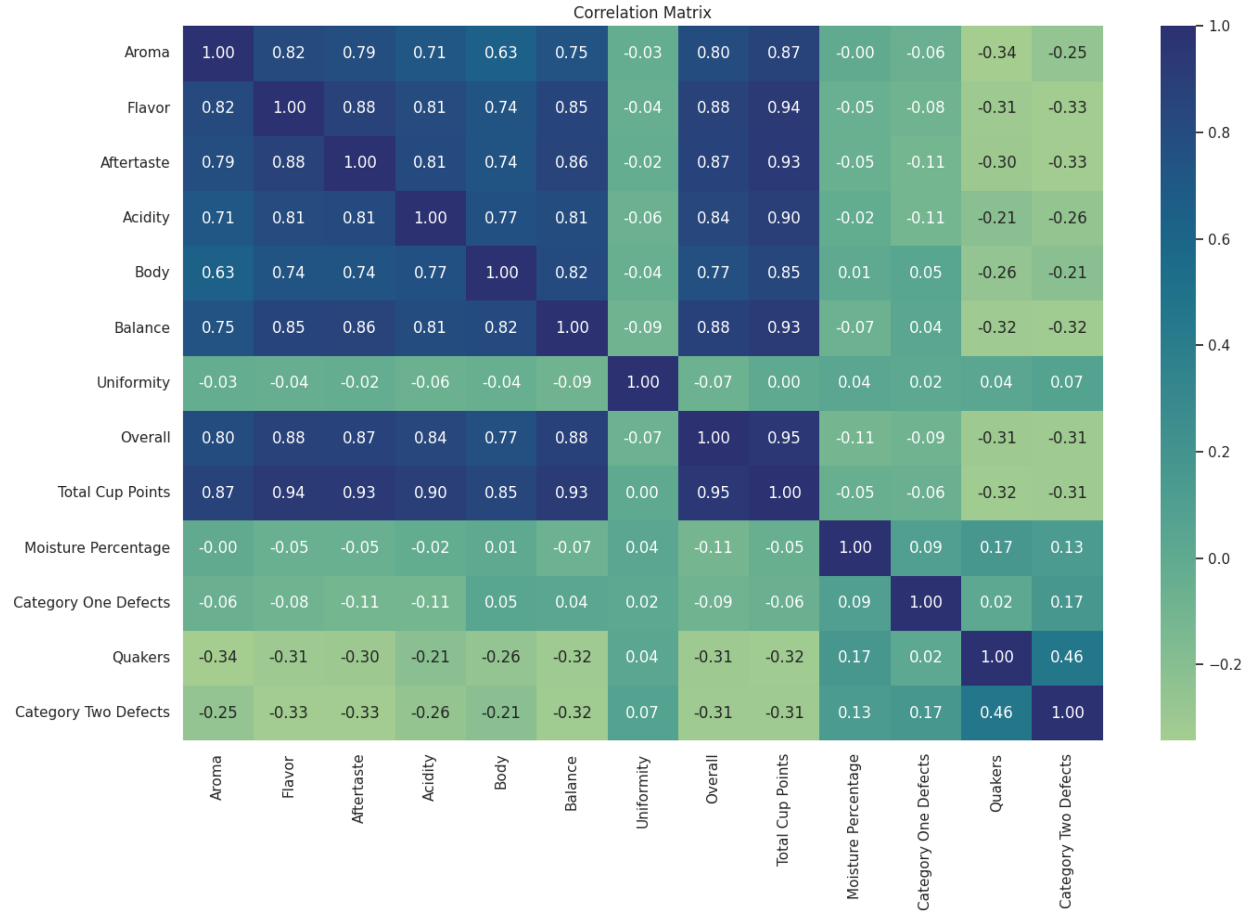
# 視覺化觀察資料

In [11]:

```
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

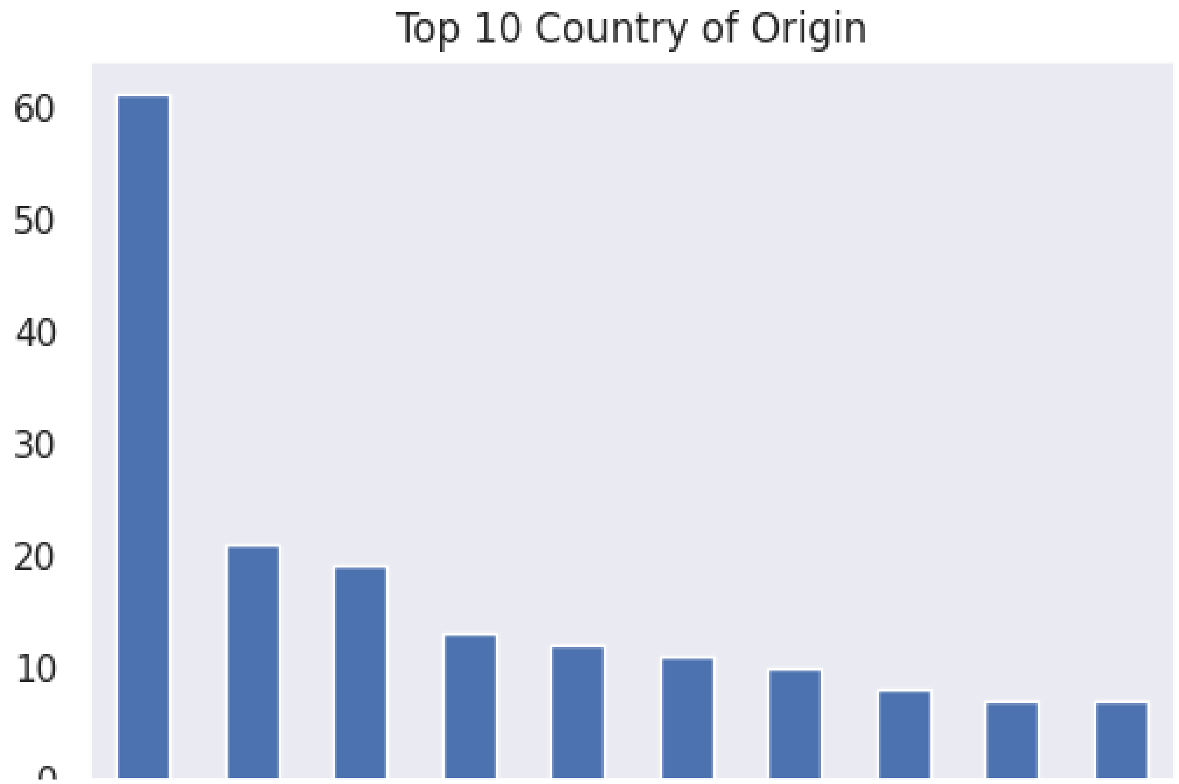
In [12]:

```
corr_matrix = df_coffee.corr()

plt.figure(figsize=(16,10))
sns.heatmap(corr_matrix, annot=True, cbar=True, fmt=".2f", cmap="crest")
plt.title("Correlation Matrix")
plt.show()
```



In [13]:

```
cat_list = df_coffee.select_dtypes(include=["object"]).columns.tolist()

for col in cat_list:
    plt.figure(figsize=(6,4))
    top10 = df_coffee[col].value_counts()[:10]
    top10.plot(kind='bar')
    plt.title("Top 10 " + col)
    plt.grid(visible=False)
    plt.show()
```



# 定義目標與切分 train、test

In [14]:

```
from sklearn.model_selection import train_test_split
```

In [15]:

```
df_coffee["Country of Origin"].value_counts()
```

Out[15]:

```
Taiwan                          61
Guatemala                       21
Colombia                        19
Honduras                        13
Thailand                        12
Ethiopia                        11
Brazil                          10
Costa Rica                       8
Nicaragua                        7
El Salvador                      7
Tanzania, United Republic Of     6
United States (Hawaii)           5
Mexico                           4
Peru                             4
Vietnam                          4
Uganda                           3
Indonesia                        3
Laos                             3
Panama                           2
Kenya                            2
Madagascar                       1
Myanmar                          1
Name: Country of Origin, dtype: int64
```

有些國家太少，這邊取 top 7 (數量 > 10)

In [16]:

```
others_country = df_coffee["Country of Origin"].value_counts()[7:].index
print('other:', others_country)
```

```
other: Index(['Costa Rica', 'Nicaragua', 'El Salvador',
       'Tanzania, United Republic Of', 'United States (Hawaii)', 'M
exico',
       'Peru', 'Vietnam', 'Uganda', 'Indonesia', 'Laos', 'Panama',
'Kenya',
       'Madagascar', 'Myanmar'],
      dtype='object')
```

In [17]:

```
df_coffee.loc[df_coffee["Country of Origin"].isin(others_country), "Country of Or
```

看一下處理完的 y 的數量

In [18]:

```
df_coffee["Country of Origin"].value_counts()
```

Out[18]:

```
Taiwan       61
others       60
Guatemala    21
Colombia     19
Honduras     13
Thailand     12
Ethiopia     11
Brazil       10
Name: Country of Origin, dtype: int64
```

切分資料

In [112]:

```
X = df_coffee.drop(labels="Country of Origin", axis=1)
y = df_coffee["Country of Origin"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

這邊要注意的是，sklearn 的 train_test_split 會根據你丟進去的 data type 去輸出對應格式的切分結果。如果我 X、y 給他 array 的格式，最後輸出的也是 array。在這邊我們直接使用 pandas dataframe 格式，他會輸出 dataframe 的格式，我就可以透過 index 去追朔切分完的資料是對應哪筆原始的資料。

In [20]:

```
X_train
```

Out[20]:

| | Variety | Processing Method | Aroma | Flavor | Aftertaste | Acidity | Body | Balance | Unif |
|---|---|---|---|---|---|---|---|---|---|
| **34** | Ethiopian Heirlooms | Natural / Dry | 8.00 | 8.08 | 8.00 | 8.00 | 7.67 | 7.75 | |
| **132** | Lempira | Washed / Wet | 7.67 | 7.75 | 7.67 | 7.58 | 7.50 | 7.58 | |
| **123** | Typica | Washed / Wet | 7.58 | 7.67 | 7.58 | 7.75 | 7.42 | 7.58 | |
| **156** | Bourbon | Washed / Wet | 7.67 | 7.33 | 7.17 | 7.67 | 7.83 | 7.50 | |
| **175** | Catimor | Washed / Wet | 7.17 | 7.50 | 7.42 | 7.42 | 7.58 | 7.50 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **140** | Typica | Pulped natural / honey | 7.50 | 7.67 | 7.58 | 7.58 | 7.50 | 7.50 | |
| **102** | Typica | Washed / Wet | 7.67 | 7.92 | 7.58 | 7.58 | 7.58 | 7.75 | |
| **64** | Catrenic | Natural / Dry | 7.92 | 7.75 | 7.67 | 7.67 | 7.83 | 7.75 | |
| **56** | Yellow Bourbon | Double Carbonic Maceration / Natural | 7.83 | 7.92 | 7.75 | 7.92 | 7.67 | 7.83 | |
| **190** | Java | Washed / Wet | 7.33 | 7.42 | 7.25 | 7.33 | 7.50 | 7.42 | |

165 rows × 16 columns

可以看到這樣就可以，透過 index 直接找到對應的資料。

In [21]:

```
df_coffee.iloc[X_train.index]
```

Out[21]:

| | Country of Origin | Variety | Processing Method | Aroma | Flavor | Aftertaste | Acidity | Body | Bal |
|---|---|---|---|---|---|---|---|---|---|
| **34** | Ethiopia | Ethiopian Heirlooms | Natural / Dry | 8.00 | 8.08 | 8.00 | 8.00 | 7.67 | |
| **132** | Honduras | Lempira | Washed / Wet | 7.67 | 7.75 | 7.67 | 7.58 | 7.50 | |
| **123** | Taiwan | Typica | Washed / Wet | 7.58 | 7.67 | 7.58 | 7.75 | 7.42 | |
| **156** | others | Bourbon | Washed / Wet | 7.67 | 7.33 | 7.17 | 7.67 | 7.83 | |
| **175** | Thailand | Catimor | Washed / Wet | 7.17 | 7.50 | 7.42 | 7.42 | 7.58 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **140** | Taiwan | Typica | Pulped natural / honey | 7.50 | 7.67 | 7.58 | 7.58 | 7.50 | |
| **102** | others | Typica | Washed / Wet | 7.67 | 7.92 | 7.58 | 7.58 | 7.58 | |
| **64** | others | Catrenic | Natural / Dry | 7.92 | 7.75 | 7.67 | 7.67 | 7.83 | |
| **56** | Brazil | Yellow Bourbon | Double Carbonic Maceration / Natural | 7.83 | 7.92 | 7.75 | 7.92 | 7.67 | |
| **190** | Thailand | Java | Washed / Wet | 7.33 | 7.42 | 7.25 | 7.33 | 7.50 | |

165 rows × 17 columns

# 資料轉換

ColumnTransformer 可以結合 sklearn pipeline 的特性，對不同 columns 建立各自的 pipeline

[sklearn.compose.ColumnTransformer — scikit-learn 1.2.2 documentation (https://scikit-learn.org/stable/modules/generated/sklearn.compose.ColumnTransformer.html)](https://scikit-learn.org/stable/modules/generated/sklearn.compose.ColumnTransformer.html)

In [22]:

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, MinMaxScaler
```

In [56]:

```
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 165 entries, 34 to 190
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Variety                165 non-null    object 
 1   Processing Method      165 non-null    object 
 2   Aroma                  165 non-null    float64
 3   Flavor                 165 non-null    float64
 4   Aftertaste             165 non-null    float64
 5   Acidity                165 non-null    float64
 6   Body                   165 non-null    float64
 7   Balance                165 non-null    float64
 8   Uniformity             165 non-null    float64
 9   Overall                165 non-null    float64
 10  Total Cup Points       165 non-null    float64
 11  Moisture Percentage    165 non-null    float64
 12  Category One Defects   165 non-null    int64  
 13  Quakers                165 non-null    int64  
 14  Color                  165 non-null    object 
 15  Category Two Defects   165 non-null    int64  
dtypes: float64(10), int64(3), object(3)
memory usage: 21.9+ KB
```

In [24]:

```
cat_list = X_train.select_dtypes(include=["object"]).columns.tolist()
num_list = X_train.select_dtypes(exclude=["object"]).columns.tolist()

full_pipeline = ColumnTransformer([
    ("num", MinMaxScaler(), num_list),
    ("category", OneHotEncoder(handle_unknown="ignore"), cat_list),
])
```

In [25]:

```
X_train_norm = full_pipeline.fit_transform(X_train)
X_test_norm = full_pipeline.transform(X_test)
X_norm = full_pipeline.transform(X)
```

自動轉換為稀疏矩陣的資料格式

In [26]:

```
X_train_norm
```

Out[26]:

```
<165x73 sparse matrix of type '<class 'numpy.float64'>'
	with 2290 stored elements in Compressed Sparse Row format>
```

In [27]:

```
print(X_train_norm)
```

```
(0, 0)        0.7211538461538463
(0, 1)        0.7599999999999998
(0, 2)        0.7599999999999998
(0, 3)        0.6685714285714286
(0, 4)        0.591549295774648
(0, 5)        0.6171428571428565
(0, 6)        1.0
(0, 7)        0.6073298429319371
(0, 8)        0.6469549867608118
(0, 9)        0.9111111111111111
(0, 11)       0.25
(0, 12)       0.3076923076923077
(0, 28)       1.0
(0, 57)       1.0
(0, 70)       1.0
(1, 0)        0.5625
(1, 1)        0.5714285714285712
(1, 2)        0.5714285714285712
(1, 3)        0.4285714285714284
(1, 4)        0.471830985915493
(1, 5)        0.5199999999999996
(1, 6)        1.0
(1, 7)        0.4345549738219896
(1, 8)        0.4633715798764344
(1, 9)        0.7777777777777777
:     :
(163, 5)      0.6628571428571428
(163, 6)      1.0
(163, 7)      0.6073298429319371
(163, 8)      0.5957634598411294
(163, 9)      0.8296296296296295
(163, 11)     0.08333333333333333
(163, 12)     0.38461538461538464
(163, 50)     1.0
(163, 55)     1.0
(163, 66)     1.0
(164, 0)      0.3990384615384617
(164, 1)      0.38285714285714256
(164, 2)      0.33142857142857096
(164, 3)      0.28571428571428603
(164, 4)      0.471830985915493
(164, 5)      0.42857142857142794
(164, 6)      1.0
(164, 7)      0.26178010471204205
(164, 8)      0.3018534863195059
(164, 9)      0.8592592592592592
(164, 10)     0.5
(164, 12)     0.07692307692307693
(164, 31)     1.0
(164, 61)     1.0
(164, 66)     1.0
```

# 模型預測: random forest

In [29]:

```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(random_state=0)
clf.fit(X_train_norm, y_train)

rf_pred = clf.predict(X_test_norm)
print(rf_pred)
```

```
['Colombia' 'others' 'others' 'others' 'Colombia' 'others' 'Guatema
la'
 'others' 'Taiwan' 'Ethiopia' 'others' 'Honduras' 'Taiwan' 'Taiwan'
 'others' 'others' 'Taiwan' 'Taiwan' 'others' 'Taiwan' 'Taiwan'
 'Guatemala' 'Taiwan' 'Taiwan' 'Colombia' 'Taiwan' 'others' 'Taiwa
n'
 'Guatemala' 'others' 'others' 'Taiwan' 'others' 'Taiwan' 'Taiwan'
 'Ethiopia' 'others' 'others' 'Taiwan' 'Taiwan' 'Taiwan' 'others']
```

In [30]:

```
from sklearn.metrics import classification_report

print(classification_report(y_test, rf_pred))
```

```
              precision    recall  f1-score   support

      Brazil       0.00      0.00      0.00         1
    Colombia       0.67      0.33      0.44         6
    Ethiopia       1.00      0.50      0.67         4
   Guatemala       0.33      0.17      0.22         6
    Honduras       1.00      0.50      0.67         2
      Taiwan       0.65      1.00      0.79        11
    Thailand       0.00      0.00      0.00         2
      others       0.50      0.80      0.62        10

    accuracy                           0.60        42
   macro avg       0.52      0.41      0.43        42
weighted avg       0.57      0.60      0.54        42
```

# 模型預測: kmeans (k=目標類別數)

In [31]:

```
from sklearn.cluster import KMeans
kmeans = KMeans(
      init="random",
      n_clusters=7,
      n_init='auto',
      random_state=42
)
kmeans.fit(X_norm)
```

Out[31]:

```
KMeans(init='random', n_clusters=7, n_init='auto', random_state=42)
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

要注意的是這邊出來的是全部的結果(train + test)，需要把對應 test 結果抽取出來才能做比較。這時候就需要使用到上面切分 train、test 時對應原始資料的 index

In [32]:

```
kmeans.labels_
```

Out[32]:

```
array([5, 5, 5, 5, 5, 5, 5, 6, 0, 0, 6, 5, 5, 5, 2, 4, 1, 5, 5, 0,
5, 6,
       4, 3, 3, 2, 5, 5, 5, 1, 3, 1, 5, 1, 6, 0, 3, 5, 0, 3, 6, 0,
5, 1,
       1, 5, 2, 1, 2, 1, 6, 3, 0, 0, 6, 1, 1, 3, 4, 2, 6, 2, 4, 5,
6, 5,
       6, 6, 5, 4, 1, 1, 6, 3, 1, 3, 6, 4, 1, 4, 1, 3, 1, 2, 4, 5,
2, 2,
       4, 0, 1, 0, 2, 2, 5, 1, 6, 2, 6, 0, 1, 2, 1, 4, 1, 0, 3, 6,
4, 6,
       6, 4, 2, 1, 3, 1, 0, 1, 1, 3, 1, 4, 1, 1, 0, 2, 3, 3, 0, 6,
1, 4,
       3, 1, 6, 3, 6, 1, 1, 3, 4, 1, 0, 3, 1, 3, 4, 6, 4, 2, 0, 6,
1, 6,
       2, 1, 1, 2, 6, 3, 3, 1, 1, 3, 3, 1, 3, 6, 0, 1, 1, 1, 3, 4,
1, 1,
       2, 2, 4, 0, 0, 4, 4, 4, 6, 3, 1, 2, 4, 4, 1, 6, 1, 0, 6, 6,
0, 1,
       0, 0, 6, 6, 6, 6, 1, 6, 1], dtype=int32)
```

In [44]:

```
kmeans_train_pred = kmeans.labels_[X_train.index]
```

需要先依照 train 的結果將 kmean 預測的 group 轉換為對應類別

In [46]:

```
map_df = pd.DataFrame({'kmeans': kmeans_train_pred, 'label': y_train})
map_df
```

Out[46]:

| | kmeans | label |
|---|---|---|
| **34** | 6 | Ethiopia |
| **132** | 3 | Honduras |
| **123** | 1 | Taiwan |
| **156** | 1 | others |
| **175** | 1 | Thailand |
| **...** | ... | ... |
| **140** | 4 | Taiwan |
| **102** | 1 | others |
| **64** | 6 | others |
| **56** | 1 | Brazil |
| **190** | 1 | Thailand |

165 rows × 2 columns

In [55]:

```
map_dt={}
for gp in map_df['kmeans'].unique():
    most_freq_label = map_df.loc[map_df['kmeans'] == gp, 'label'].value_counts().
    map_dt.update({gp: most_freq_label})
```

可以看到各 group 對應的標籤，有些標籤可能沒有，這是正常的。

In [54]:

```
map_dt
```

Out[54]:

```
{6: 'Taiwan',
 3: 'others',
 1: 'others',
 2: 'Guatemala',
 0: 'others',
 4: 'Taiwan',
 5: 'Taiwan'}
```

In [66]:

```
kmeans_test_pred = kmeans.labels_[X_test.index]
kmeans_test_pred
```

Out[66]:

```
array([2, 2, 1, 6, 1, 6, 1, 6, 3, 6, 2, 1, 4, 1, 4, 4, 3, 6, 6, 0,
5, 6,
       1, 4, 0, 0, 3, 5, 5, 1, 4, 4, 4, 5, 5, 0, 2, 0, 5, 5, 3, 0],
      dtype=int32)
```

In [67]:

```
kmeans_test_pred = np.array([map_dt[i] for i in kmeans_test_pred])
kmeans_test_pred
```

Out[67]:

```
array(['Guatemala', 'Guatemala', 'others', 'Taiwan', 'others', 'Tai
wan',
       'others', 'Taiwan', 'others', 'Taiwan', 'Guatemala', 'other
s',
       'Taiwan', 'others', 'Taiwan', 'Taiwan', 'others', 'Taiwan',
       'Taiwan', 'others', 'Taiwan', 'Taiwan', 'others', 'Taiwan',
       'others', 'others', 'others', 'Taiwan', 'Taiwan', 'others',
       'Taiwan', 'Taiwan', 'Taiwan', 'Taiwan', 'Taiwan', 'others',
       'Guatemala', 'others', 'Taiwan', 'Taiwan', 'others', 'other
s'],
      dtype='<U9')
```

轉換完成後就可以評估我們 kmeans 的結果

In [68]:

```
from sklearn.metrics import classification_report

print(classification_report(y_test, kmeans_test_pred))
```

```
              precision    recall  f1-score   support

      Brazil       0.00      0.00      0.00         1
    Colombia       0.00      0.00      0.00         6
    Ethiopia       0.00      0.00      0.00         4
   Guatemala       0.25      0.17      0.20         6
    Honduras       0.00      0.00      0.00         2
      Taiwan       0.33      0.64      0.44        11
    Thailand       0.00      0.00      0.00         2
      others       0.24      0.40      0.30        10

    accuracy                           0.29        42
   macro avg       0.10      0.15      0.12        42
weighted avg       0.18      0.29      0.21        42
```

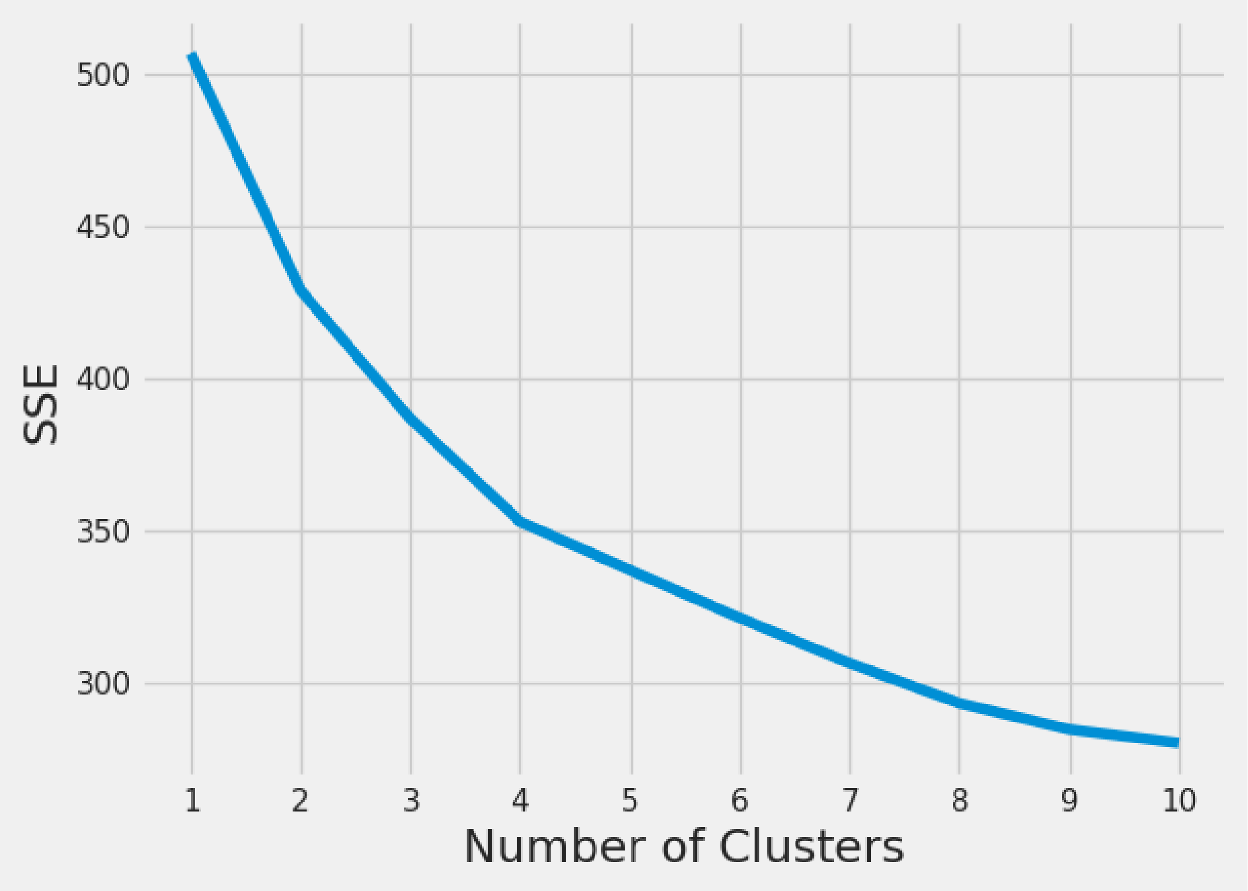
# 模型預測: kmeans (elbow 選擇 k)

In [75]:

```
from sklearn.cluster import KMeans

kmeans_kwargs = {
    "init": "random",
    "random_state": 42,
}

# A list holds the SSE values for each k
sse = []
for k in range(1, 11):
    kmeans = KMeans(
        n_clusters=k,
        init="random",
        n_init='auto',
        random_state=42
        )
    kmeans.fit(X_norm)
    sse.append(kmeans.inertia_)
```

In [76]:

```
plt.style.use("fivethirtyeight")
plt.plot(range(1, 11), sse)
plt.xticks(range(1, 11))
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.show()
```



In [72]:

```
!pip install kneed
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simp
le,) https://us-python.pkg.dev/colab-wheels/public/simple/ (http
s://us-python.pkg.dev/colab-wheels/public/simple/)
Collecting kneed
  Downloading kneed-0.8.3-py3-none-any.whl (10 kB)
Requirement already satisfied: numpy>=1.14.2 in /usr/local/lib/pyth
on3.10/dist-packages (from kneed) (1.22.4)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/pytho
n3.10/dist-packages (from kneed) (1.10.1)
Installing collected packages: kneed
Successfully installed kneed-0.8.3
```

In [77]:

```
from kneed import KneeLocator

kl = KneeLocator(
    range(1, 11), sse, curve="convex", direction="decreasing"
)

kl.elbow
```

Out[77]:

```
4
```

使用 k = 4

In [78]:

```
kmeans = KMeans(
    n_clusters=4,
    init="random",
    n_init='auto',
    random_state=42
    )
kmeans.fit(X_norm)
```

Out[78]:

```
KMeans(init='random', n_clusters=4, n_init='auto', random_state=42)
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [79]:

```
kmeans_train_pred = kmeans.labels_[X_train.index]
map_df = pd.DataFrame({'kmeans': kmeans_train_pred, 'label': y_train})

map_dt={}
for gp in map_df['kmeans'].unique():
    most_freq_label = map_df.loc[map_df['kmeans'] == gp, 'label'].value_counts().
    map_dt.update({gp: most_freq_label})

kmeans_test_pred = kmeans.labels_[X_test.index]
kmeans_test_pred = np.array([map_dt[i] for i in kmeans_test_pred])
kmeans_test_pred
```

Out[79]:

```
array(['others', 'others', 'others', 'Taiwan', 'others', 'Taiwan',
       'others', 'Taiwan', 'others', 'Taiwan', 'others', 'others',
       'Taiwan', 'others', 'Taiwan', 'Taiwan', 'others', 'Taiwan',
       'Taiwan', 'others', 'others', 'Taiwan', 'others', 'Taiwan',
       'others', 'others', 'others', 'Taiwan', 'Taiwan', 'others',
       'Taiwan', 'Taiwan', 'Taiwan', 'Taiwan', 'others', 'others',
       'others', 'others', 'Taiwan', 'others', 'others', 'others'],
      dtype='<U6')
```

In [80]:

```
from sklearn.metrics import classification_report

print(classification_report(y_test, kmeans_test_pred))
```

```
              precision    recall  f1-score   support

      Brazil       0.00      0.00      0.00         1
    Colombia       0.00      0.00      0.00         6
    Ethiopia       0.00      0.00      0.00         4
   Guatemala       0.00      0.00      0.00         6
    Honduras       0.00      0.00      0.00         2
      Taiwan       0.22      0.36      0.28        11
    Thailand       0.00      0.00      0.00         2
      others       0.21      0.50      0.29        10

    accuracy                           0.21        42
   macro avg       0.05      0.11      0.07        42
weighted avg       0.11      0.21      0.14        42
```

## 模型預測: random forest + kmeans

In [81]:

```
kmeans = KMeans(
    n_clusters=4,
    init="random",
    n_init='auto',
    random_state=42
    )
kmeans.fit(X_norm)
```

Out[81]:

```
KMeans(init='random', n_clusters=4, n_init='auto', random_state=42)
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [93]:

```
kmeans_train_pred = kmeans.labels_[X_train.index]
X_train['kmean_label'] = kmeans_train_pred
```

In [94]:

```
kmeans_test_pred = kmeans.labels_[X_test.index]
X_test['kmean_label'] = kmeans_test_pred
```

這邊直接用上面 資料轉換 地方定義的 full pipeline 一樣的方式進行正規化、onehot encoding

In [89]:

```
cat_list.append('kmean_label')
cat_list
```

Out[89]:

```
['Variety', 'Processing Method', 'Color', 'kmean_label']
```

In [95]:

```
full_pipeline_kmean = ColumnTransformer([
    ("num", MinMaxScaler(), num_list),
    ("category", OneHotEncoder(handle_unknown="ignore"), cat_list),
])
```

In [96]:

```
X_train_kmean_norm = full_pipeline_kmean.fit_transform(X_train)
X_test_kmean_norm = full_pipeline_kmean.transform(X_test)
```

In [99]:

```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(random_state=0)
clf.fit(X_train_kmean_norm, y_train)

rf_kmean_pred = clf.predict(X_test_kmean_norm)
```

In [100]:

```
from sklearn.metrics import classification_report

print(classification_report(y_test, rf_kmean_pred))
```

```
              precision    recall  f1-score   support

      Brazil       0.00      0.00      0.00         1
    Colombia       0.67      0.33      0.44         6
    Ethiopia       1.00      0.50      0.67         4
   Guatemala       0.33      0.17      0.22         6
    Honduras       1.00      0.50      0.67         2
      Taiwan       0.67      0.91      0.77        11
    Thailand       0.00      0.00      0.00         2
      others       0.50      0.90      0.64        10

    accuracy                           0.60        42
   macro avg       0.52      0.41      0.43        42
weighted avg       0.58      0.60      0.55        42
```