Generic-food Dataset

Importing the libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Importing the dataset

```
dataset = pd.read_csv("generic-food.csv")
dataset.shape
dataset.tail()
```

	FOOD NAME	SCIENTIFIC NAME	GROUP	SUB GROUP
918	Whelk	Buccinidae	Aquatic foods	Mollusks
919	Coalfish pollock	Pollachius virens	Aquatic foods	Fishes
920	Broad whitefish	Coregonus nasus	Aquatic foods	Fishes
921	Whitefish	Coregonus	Aquatic foods	Fishes
922	Whiting	Merlangius merlangus	Aquatic foods	Fishes

Handling Missing data

```
dataset.isnull().values.any()
```

```
True
```

```
dataset.isnull()
```

	FOOD NAME	SCIENTIFIC NAME	GROUP	SUB GROUP
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	False	False
...	...	...	...	...
918	False	False	False	False
919	False	False	False	False
920	False	False	False	False
921	False	False	False	False
922	False	False	False	False

923 rows × 4 columns

```
dataset.isnull().sum()
```

```
FOOD NAME            0
SCIENTIFIC NAME    259
GROUP                0
SUB GROUP            0
dtype: int64
```

```
dataset.dropna(inplace=True)
dataset.shape
```

```
(664, 4)
```

Handling Duplicate Data

```
dataset.duplicated(subset=None, keep=False).value_counts()
```

```
False    632
True      32
dtype: int64
```

```
bool_series = dataset.duplicated(subset=None, keep=False)
df = dataset[~bool_series]
```

```
print("Before removing duplicates:")
print(dataset.shape)
print("After removing duplicate tuples:")
print(df.shape)
```

```
Before removing duplicates:
(664, 4)
After removing duplicate tuples:
(632, 4)
```

DATA NORMALIZATION

```
from sklearn import preprocessing
```

```
X = df.iloc[:,2:4]
```

```
print(X)
```

```
                 GROUP                SUB GROUP
0    Herbs and Spices                    Herbs
1          Vegetables                 Cabbages
2    Herbs and Spices                    Herbs
3              Fruits          Tropical fruits
4          Vegetables  Onion-family vegetables
..                ...                     ...
898            Fruits                  Berries
899            Fruits                  Berries
900            Gourds                   Gourds
901        Vegetables                 Cabbages
902        Vegetables          Leaf vegetables

[632 rows x 2 columns]
```

```
Y = df.iloc[:,0:2]
```

```
print(Y)
```

```
              FOOD NAME                                    SCIENTIFIC NAME
0             Angelica                                    Angelica keiskei
1        Savoy cabbage                 Brassica oleracea var. sabauda
2        Silver linden                                    Tilia argentea
3                 Kiwi                                 Actinidia chinensis
4       Allium (Onion)                                           Allium
..                 ...                                              ...
898    Saskatoon berry                              Amelanchier alnifolia
899     Nanking cherry                                   Prunus tomentosa
900   Japanese pumpkin                                   Cucurbita maxima
901      White cabbage  Brassica oleracea L. var. capitata L. f. alba DC.
902    Romaine lettuce                Lactuca sativa L. var. longifolia

[632 rows x 2 columns]
```

```
le = preprocessing.LabelEncoder()
X = X.apply(le.fit_transform).head()
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
```

```
print(X)
```

```
[[-0.55738641 -0.34470465]
 [ 1.18444612 -1.30987765]
 [-0.55738641 -0.34470465]
 [-1.25411943  1.72352323]
 [ 1.18444612  0.27576372]]
```

```
#zscore normalization
from scipy import stats
X = stats.zscore(X)
```

```
print(X)
```

```
[[-0.55738641 -0.34470465]
 [ 1.18444612 -1.30987765]
 [-0.55738641 -0.34470465]
 [-1.25411943  1.72352323]
 [ 1.18444612  0.27576372]]
```

```
#Min max Normalization
from sklearn.preprocessing import MinMaxScaler
min_max_scaler = MinMaxScaler()
X = min_max_scaler.fit_transform(X)
```

```
print(X)
```

```
[[0.28571429 0.31818182]
 [1.         0.        ]
 [0.28571429 0.31818182]
 [0.         1.        ]
 [1.         0.52272727]]
```

Data Transformation

Encoding the categorical data in Group and SubGroup column using OneHotEncoder

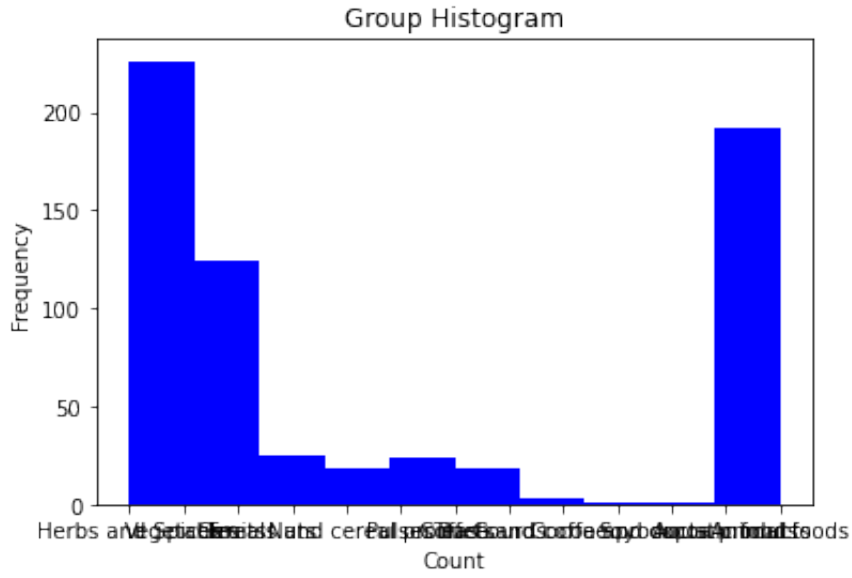
one hot encoder for group and sub group

```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()
enc.fit_transform(Y).toarray()
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

HISTOGRAM

```
plt.hist(df['GROUP'],color='blue',orientation='vertical')
plt.title('Group Histogram')
plt.xlabel('Count')
plt.ylabel('Frequency')
plt.show()
```




```
plt.hist(df['SUB GROUP'],color='red',orientation='vertical')
plt.title('SUB Group Histogram')
plt.xlabel('Count')
plt.ylabel('Frequency')
plt.show()
```