In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
df=pd.read_csv("imports-by-country-breed-type-age-and-sex-april-2009-1.csv")
```

In [3]:

```
df.head()
```

Out[3]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Country	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	U
0	Age	Breed Type	Sex	AUSTRIA	CZECH REPUBLIC	DENMARK	FRANCE	GERMANY	
1	01. Under 1	Non Dairy	F	0	0	0	0	0	
2	NaN	NaN	M	0	0	0	0	0	
3	02. 1 to 2	Non Dairy	F	0	0	0	0	0	
4	NaN	NaN	M	0	0	0	0	0	

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46 entries, 0 to 45
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Unnamed: 0   16 non-null     object
 1   Unnamed: 1   27 non-null     object
 2   Unnamed: 2   45 non-null     object
 3   Country      46 non-null     object
 4   Unnamed: 4   46 non-null     object
 5   Unnamed: 5   46 non-null     object
 6   Unnamed: 6   46 non-null     object
 7   Unnamed: 7   46 non-null     object
 8   Unnamed: 8   46 non-null     object
 9   Unnamed: 9   46 non-null     object
 10  Unnamed: 10  46 non-null     object
 11  Unnamed: 11  46 non-null     object
 12  Total        45 non-null     object
dtypes: object(13)
memory usage: 4.8+ KB
```

In [5]:

```
df.describe()
```

Out[5]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Country	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed 7
count	16	27	45	46	46	46	46	46
unique	16	3	3	5	5	5	9	8
top	Age	Non Dairy	F	0	0	0	0	0
freq	1	14	22	42	42	42	36	39

In [6]:

```
e","Sex","AUSTRIA","CZECH R-EPUBLIC","DENMARK","FRANCE","GERMANY","IRELAND","NETHERLANDS","N
```

In [7]:

```
df.head()
```

Out[7]:

	Age	Breed Type	Sex	AUSTRIA	CZECH R-EPUBLIC	DENMARK	FRANCE	GERMANY	IRELAND	NETH
0	Age	Breed Type	Sex	AUSTRIA	CZECH REPUBLIC	DENMARK	FRANCE	GERMANY	IRELAND	NETH
1	01. Under 1	Non Dairy	F	0	0	0	0	0	24	
2	NaN	NaN	M	0	0	0	0	0	0	
3	02. 1 to 2	Non Dairy	F	0	0	0	0	0	18	
4	NaN	NaN	M	0	0	0	0	0	0	

In [8]:

```
df.drop(0,axis=0,inplace=True)
```

In [9]:

```
df.drop(["Age"],axis=1,inplace=True)
```

In [10]:

```
df.isna().sum()
```

Out[10]:

```
Breed Type          19
Sex                  1
AUSTRIA              0
CZECH R-EPUBLIC      0
DENMARK              0
FRANCE               0
GERMANY              0
IRELAND              0
NETHERLANDS          0
NORTHERN IRELAND     0
SWEDEN               0
Total                0
dtype: int64
```

In [11]:

```
df["Breed Type"].value_counts()
```

Out[11]:

```
Non Dairy    14
Dairy        12
Name: Breed Type, dtype: int64
```

In [12]:

```
from sklearn.impute import SimpleImputer
```

In [13]:

```
si=SimpleImputer(missing_values=np.nan,strategy="constant")
df["Breed Type"]=si.fit_transform(df[["Breed Type"]])
```

In [14]:

```
df.Sex.fillna("F",inplace=True)
```

In [15]:

```
df.isna().sum()
```

Out[15]:

```
Breed Type          0
Sex                 0
AUSTRIA             0
CZECH R-EPUBLIC     0
DENMARK             0
FRANCE              0
GERMANY             0
IRELAND             0
NETHERLANDS         0
NORTHERN IRELAND    0
SWEDEN              0
Total               0
dtype: int64
```

In [16]:

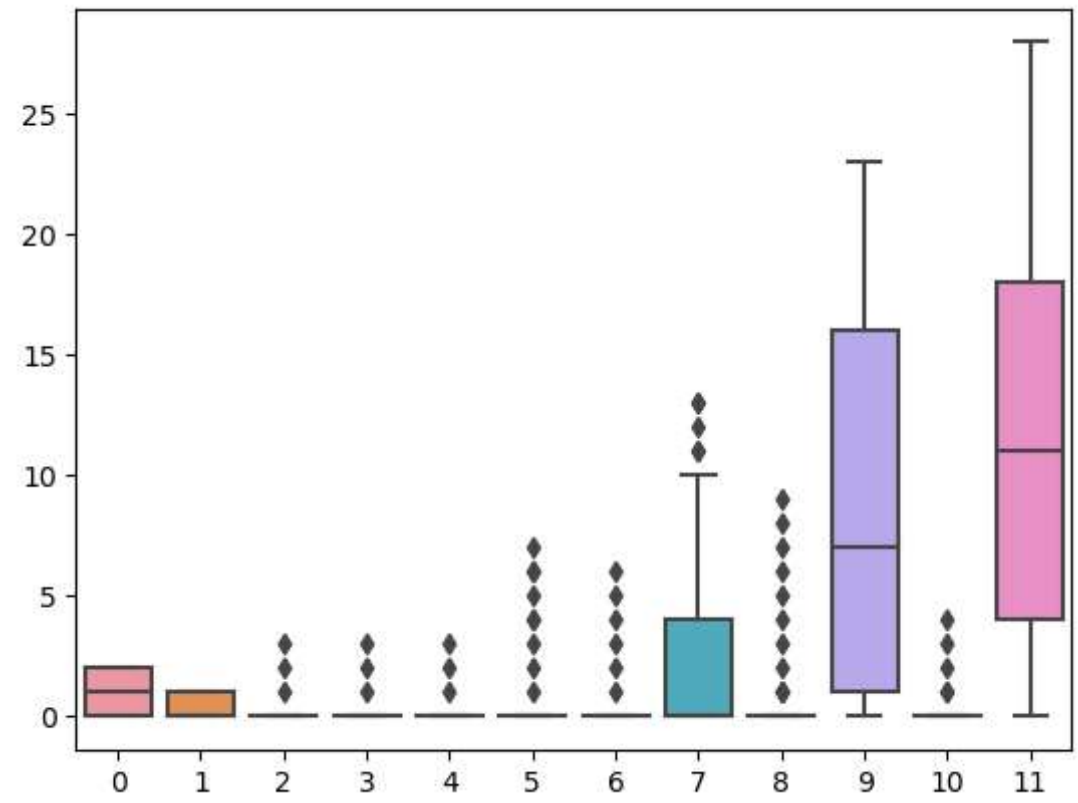
```
from sklearn.preprocessing import OrdinalEncoder
```

In [17]:

```
one=OrdinalEncoder()
df=one.fit_transform(df)
df=pd.DataFrame(df)
```

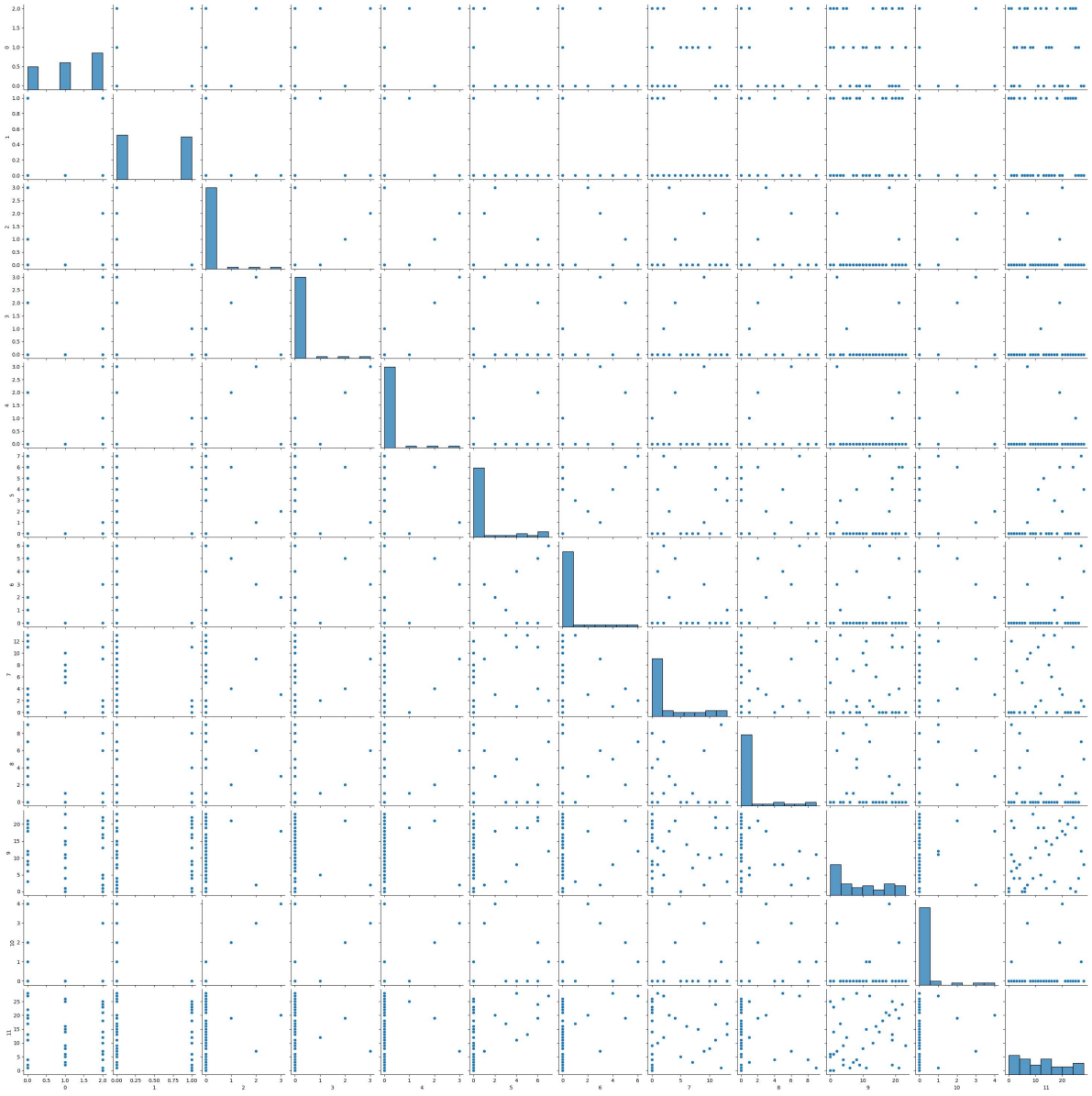
In [18]:

```
sns.boxplot(data=df);
```



In [21]:

```
sns.pairplot(df);
```



In [35]:

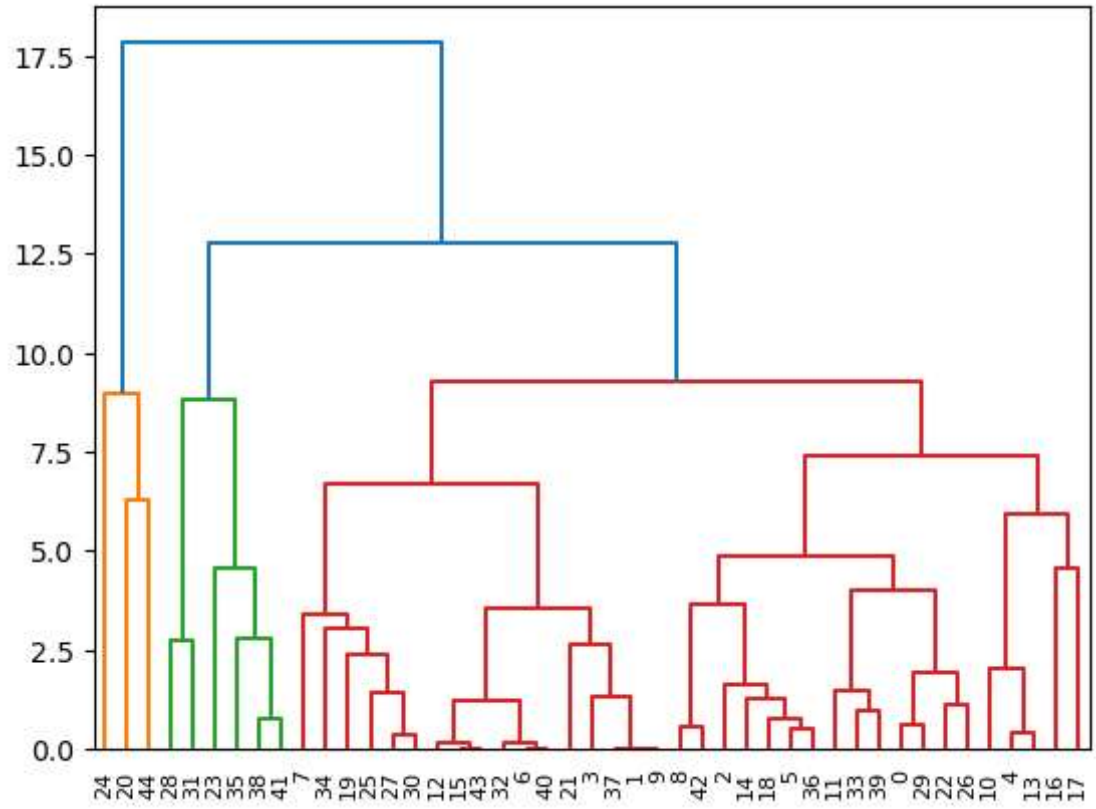
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
df = sc.fit_transform(df)
df=pd.DataFrame(df)
```

In [36]:

```
from scipy.cluster import hierarchy as hi
```

In [37]:

```
a = hi.linkage(df, method="ward")
b = hi.dendrogram(a)
```



In [38]:

```
from sklearn.cluster import AgglomerativeClustering
c = AgglomerativeClustering(n_clusters=3)
ylabel = c.fit_predict(df)
```

In [39]:

```
ylabel
```

Out[39]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 2, 1, 0, 0, 0, 2, 0, 0, 2, 0, 0, 0, 2, 0, 0, 2, 0, 0, 2, 0, 0,
       1], dtype=int64)
```

In [40]:

```
df["target"]=ylabel
```

In [41]:

```
df["target"].value_counts()
```

Out[41]:

```
0    36
2     6
1     3
Name: target, dtype: int64
```

In [42]:

```
df.head(3)
```

Out[42]:

	0	1	2	3	4	5	6	7	
0	-0.190799	-0.978019	-0.246183	-0.246183	-0.246183	-0.447959	-0.347404	1.271818	-0.454
1	1.035765	1.022475	-0.246183	-0.246183	-0.246183	-0.447959	-0.347404	-0.620143	-0.454
2	-0.190799	-0.978019	-0.246183	-0.246183	-0.246183	-0.447959	-0.347404	0.562333	-0.454

In [43]:

```
features=df.iloc[:,:-1]
```

In [45]:

```
y=df["target"]
```

In [46]:

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(features,y, test_size=0.3, random_state=1)
```

In [47]:

```
from sklearn.naive_bayes import MultinomialNB, GaussianNB, BernoulliNB
```

In [49]:

```
def pro(one):
    one.fit(xtrain,ytrain)
    ypred=one.predict(xtest)

    train=one.score(xtrain,ytrain)
    test=one.score(xtest,ytest)
    print(f"Trainacc:{train}\nTestacc:{test}\n\n")
    print(classification_report(ytest,ypred))
    return one
```

In [56]:

```
from sklearn.metrics import classification_report
from sklearn.naive_bayes import MultinomialNB, GaussianNB, BernoulliNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
```

In [53]:

```
bnb=pro(BernoulliNB())
```

```
Trainacc:1.0
Testacc:0.8571428571428571

              precision    recall  f1-score   support

           0       0.83      1.00      0.91        10
           1       1.00      1.00      1.00         1
           2       1.00      0.33      0.50         3

    accuracy                           0.86        14
   macro avg       0.94      0.78      0.80        14
weighted avg       0.88      0.86      0.83        14
```

In [54]:

```
lr=pro(LogisticRegression())
```

```
Trainacc:1.0
Testacc:1.0

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         1
           2       1.00      1.00      1.00         3

    accuracy                           1.00        14
   macro avg       1.00      1.00      1.00        14
weighted avg       1.00      1.00      1.00        14
```

In [57]:

```
kn=pro(KNeighborsClassifier())
```

```
Trainacc:0.8709677419354839
Testacc:0.7142857142857143
```

	precision	recall	f1-score	support
0	0.71	1.00	0.83	10
1	0.00	0.00	0.00	1
2	0.00	0.00	0.00	3
accuracy			0.71	14
macro avg	0.24	0.33	0.28	14
weighted avg	0.51	0.71	0.60	14

In [58]:

```
dt=pro(DecisionTreeClassifier())
```

```
Trainacc:1.0
Testacc:1.0
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	1
2	1.00	1.00	1.00	3
accuracy			1.00	14
macro avg	1.00	1.00	1.00	14
weighted avg	1.00	1.00	1.00	14

In [ ]:

```

```