Supervised and Unsupervised Algorithm on Codon Usage Frequency

In [1]:
```
# importing required libraries
import numpy as np
import pandas as pd
import os
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import metrics

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

from sklearn.preprocessing import StandardScaler, MinMaxScaler
import pandas_profiling
from sklearn.metrics import roc_auc_score, roc_curve, classification_report,accuracy_score


from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
from sklearn.manifold import TSNE
```

In [16]:
```
import warnings
warnings.filterwarnings('ignore')
```

loading the dataset

In [17]:
```
dataframe = pd.read_csv("codon_usage.csv")
```

In [18]:
```
dataframe.head()
```

Out[18]:

	Kingdom	DNAtype	SpeciesID	Ncodons	SpeciesName	UUU	UUC	UUA	UUG	CUU	...	CGG	AGA	AGG	GAU	GAC	GA
0	vrl	0	100217	1995	Epizootic haematopoietic necrosis virus	0.01654	0.01203	0.00050	0.00351	0.01203	...	0.00451	0.01303	0.03559	0.01003	0.04612	0.0120
1	vrl	0	100220	1474	Bohle iridovirus	0.02714	0.01357	0.00068	0.00678	0.00407	...	0.00136	0.01696	0.03596	0.01221	0.04545	0.0156
2	vrl	0	100755	4862	Sweet potato leaf curl virus	0.01974	0.0218	0.01357	0.01543	0.00782	...	0.00596	0.01974	0.02489	0.03126	0.02036	0.0224
3	vrl	0	100880	1915	Northern cereal mosaic virus	0.01775	0.02245	0.01619	0.00992	0.01567	...	0.00366	0.01410	0.01671	0.03760	0.01932	0.0302
4	vrl	0	100887	22831	Soil-borne cereal mosaic virus	0.02816	0.01371	0.00767	0.03679	0.01380	...	0.00604	0.01494	0.01734	0.04148	0.02483	0.0335

5 rows × 69 columns

first we shall look what all columns are available in the dataset

In [19]:
```
dataframe.columns
```

Out[19]:
```
Index(['Kingdom', 'DNAtype', 'SpeciesID', 'Ncodons', 'SpeciesName', 'UUU',
       'UUC', 'UUA', 'UUG', 'CUU', 'CUC', 'CUA', 'CUG', 'AUU', 'AUC', 'AUA',
       'AUG', 'GUU', 'GUC', 'GUA', 'GUG', 'GCU', 'GCC', 'GCA', 'GCG', 'CCU',
       'CCC', 'CCA', 'CCG', 'UGG', 'GGU', 'GGC', 'GGA', 'GGG', 'UCU', 'UCC',
       'UCA', 'UCG', 'AGU', 'AGC', 'ACU', 'ACC', 'ACA', 'ACG', 'UAU', 'UAC',
       'CAA', 'CAG', 'AAU', 'AAC', 'UGU', 'UGC', 'CAU', 'CAC', 'AAA', 'AAG',
       'CGU', 'CGC', 'CGA', 'CGG', 'AGA', 'AGG', 'GAU', 'GAC', 'GAA', 'GAG',
       'UAA', 'UAG', 'UGA'],
      dtype='object')
```

let us check any availabilities of null values and information of the dataset.

In [20]:
```
dataframe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13028 entries, 0 to 13027
Data columns (total 69 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Kingdom      13028 non-null  object 
 1   DNAtype      13028 non-null  int64  
 2   SpeciesID    13028 non-null  int64  
 3   Ncodons      13028 non-null  int64  
 4   SpeciesName  13028 non-null  object 
 5   UUU          13028 non-null  object 
 6   UUC          13028 non-null  object 
 7   UUA          13028 non-null  float64
 8   UUG          13028 non-null  float64
 9   CUU          13028 non-null  float64
 10  CUC          13028 non-null  float64
 11  CUA          13028 non-null  float64
 12  CUG          13028 non-null  float64
 13  AUU          13028 non-null  float64
 14  AUC          13028 non-null  float64
 15  AUA          13028 non-null  float64
 16  AUG          13028 non-null  float64
 17  GUU          13028 non-null  float64
 18  GUC          13028 non-null  float64
 19  GUA          13028 non-null  float64
 20  GUG          13028 non-null  float64
 21  GCU          13028 non-null  float64
 22  GCC          13028 non-null  float64
 23  GCA          13028 non-null  float64
 24  GCG          13028 non-null  float64
 25  CCU          13028 non-null  float64
 26  CCC          13028 non-null  float64
 27  CCA          13028 non-null  float64
 28  CCG          13028 non-null  float64
 29  UGG          13028 non-null  float64
 30  GGU          13028 non-null  float64
 31  GGC          13028 non-null  float64
 32  GGA          13028 non-null  float64
 33  GGG          13028 non-null  float64
 34  UCU          13028 non-null  float64
 35  UCC          13028 non-null  float64
 36  UCA          13028 non-null  float64
 37  UCG          13028 non-null  float64
 38  AGU          13028 non-null  float64
 39  AGC          13028 non-null  float64
 40  ACU          13028 non-null  float64
 41  ACC          13028 non-null  float64
 42  ACA          13028 non-null  float64
 43  ACG          13028 non-null  float64
 44  UAU          13028 non-null  float64
 45  UAC          13028 non-null  float64
 46  CAA          13028 non-null  float64
 47  CAG          13028 non-null  float64
 48  AAU          13028 non-null  float64
 49  AAC          13028 non-null  float64
 50  UGU          13028 non-null  float64
 51  UGC          13028 non-null  float64
 52  CAU          13028 non-null  float64
 53  CAC          13028 non-null  float64
 54  AAA          13028 non-null  float64
 55  AAG          13028 non-null  float64
 56  CGU          13028 non-null  float64
 57  CGC          13028 non-null  float64
 58  CGA          13028 non-null  float64
 59  CGG          13028 non-null  float64
 60  AGA          13028 non-null  float64
 61  AGG          13028 non-null  float64
 62  GAU          13028 non-null  float64
 63  GAC          13028 non-null  float64
 64  GAA          13028 non-null  float64
 65  GAG          13028 non-null  float64
 66  UAA          13028 non-null  float64
 67  UAG          13028 non-null  float64
 68  UGA          13028 non-null  float64
dtypes: float64(62), int64(3), object(4)
memory usage: 6.9+ MB
```

Now we will describe the data and let us analaze the Kingdom frequency count

#first we shall replace the short cut values in kingdom into its actual names as per the dataset description.

In [21]:
```
data.isnull().sum().sort_values(ascending=False)[:5]
```

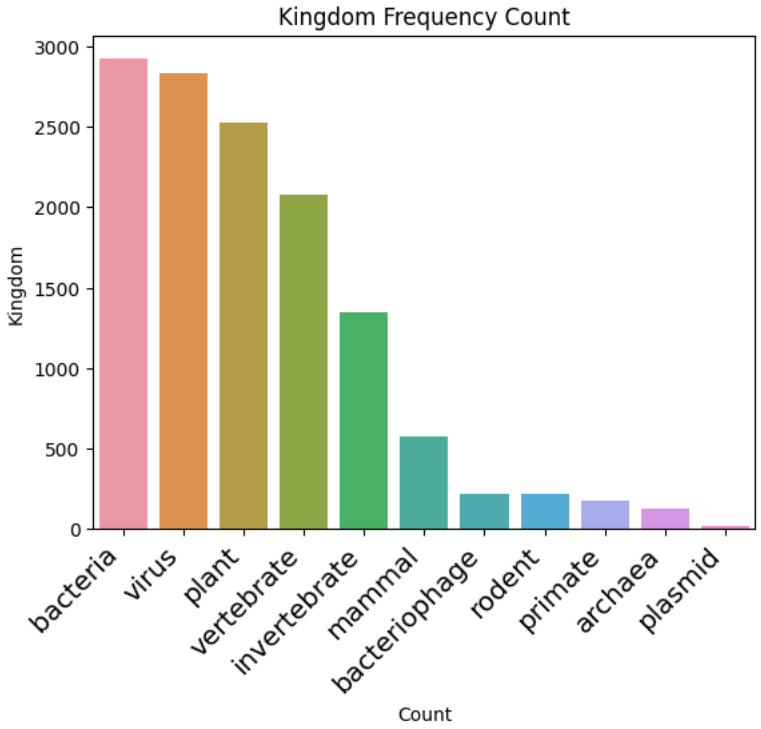
In [23]:
```
column1_names = {
                    'arc':'archaea',
                    'bct': ' bacteria',
                    'phg' : 'bacteriophage',
                    'plm': 'plasmid',
                    'pln': 'plant',
                    'inv': 'invertebrate',
                    'vrt': 'vertebrate',
                    'mam': 'mammal',
                    'rod': 'rodent',
                    'pri': 'primate',
                    'vrl': 'virus'
                }

dataframe1 = dataframe.replace({"Kingdom":column1_names})
dataframe['Kingdom_values'] = dataframe1['Kingdom']
```

In [32]:
```
sns.barplot(y = "Kingdom_values", x = "index", data = dataframe['Kingdom_values'].value_counts().reset_index())

plt.xlabel("Count")
plt.ylabel("Kingdom")
plt.title("Frequency count")
plt.xticks(
    rotation=45, horizontalalignment="right", fontweight="light", fontsize="x-large"
)

plt.show()
```



DNAtype

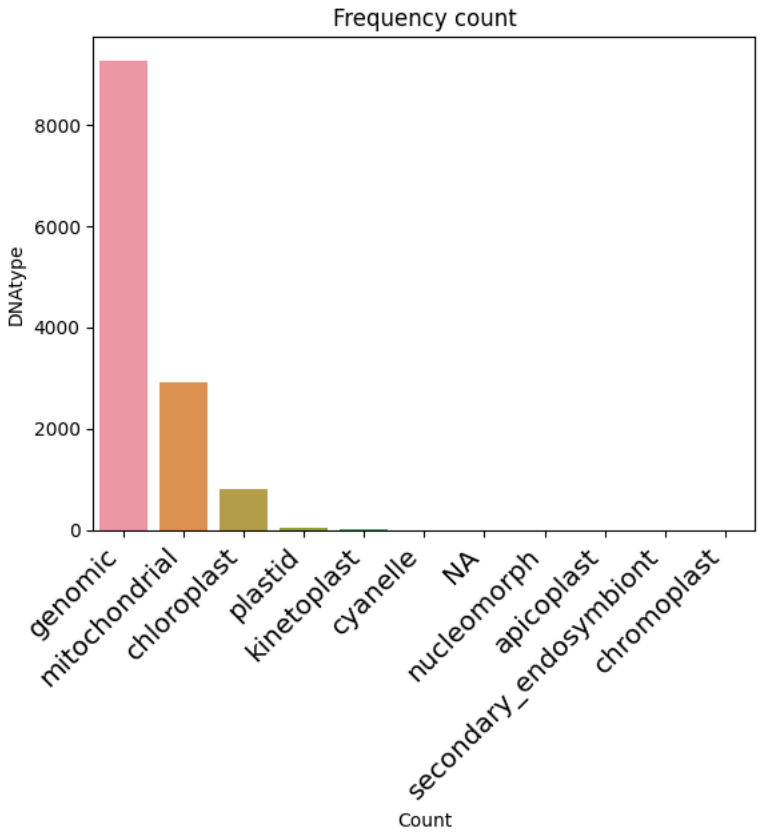
In [33]:
```
column2_names = {
                    0:'genomic',
                    1:'mitochondrial',
                    2:'chloroplast',
                    3:'cyanelle',
                    4:'plastid',
                    5:'nucleomorph',
                    6:'secondary_endosymbiont',
                    7:'chromoplast', '8':'leucoplast',
                    9:'NA',
                    10:'proplastid',
                    11:'apicoplast',
                    12:'kinetoplast'
                }

dataframe1 = dataframe.replace({"DNAtype":column2_names})
dataframe['DNAtype_values'] = dataframe1['DNAtype']
```

In [34]:
```
sns.barplot(y = "DNAtype_values", x = "index", data = dataframe['DNAtype_values'].value_counts().reset_index())

plt.xlabel("Count")
plt.ylabel("DNAtype")
plt.title("Frequency count")
plt.xticks(
    rotation=45, horizontalalignment="right", fontweight="light", fontsize="x-large"
)

plt.show()
```



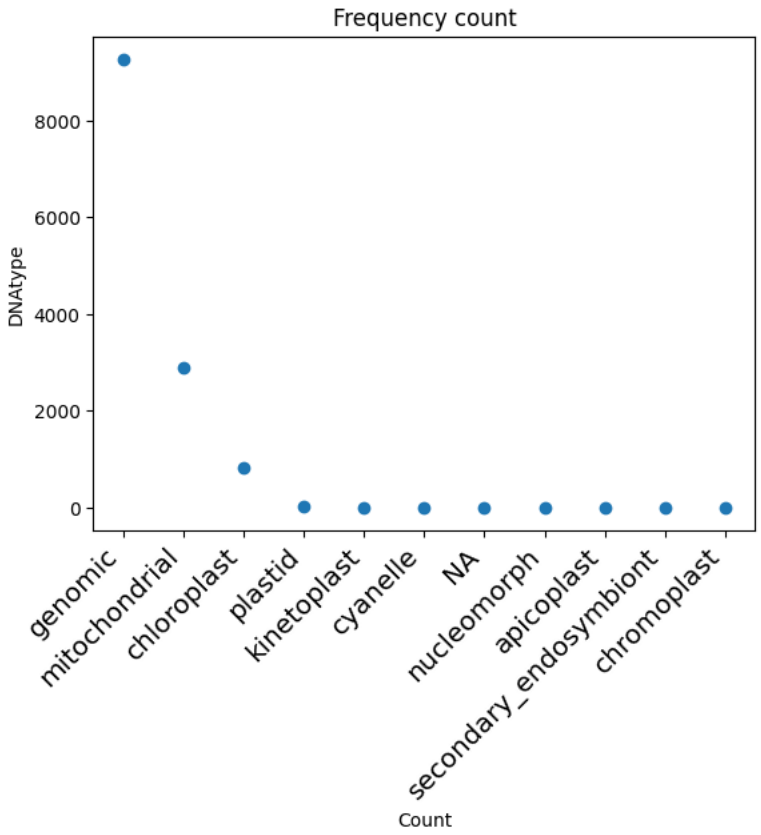
In [35]:
```
display(dataframe['Kingdom_values'].value_counts())
```

```
 bacteria        2920
virus            2832
plant            2523
vertebrate       2077
invertebrate     1345
mammal            572
bacteriophage     220
rodent            215
primate           180
archaea           126
plasmid            18
Name: Kingdom_values, dtype: int64
```

In [37]:
```
display(dataframe['DNAtype_values'].value_counts())
```

```
genomic                  9267
mitochondrial            2899
chloroplast               816
plastid                    31
kinetoplast                 5
cyanelle                    2
NA                          2
nucleomorph                 2
apicoplast                  2
secondary_endosymbiont      1
chromoplast                 1
Name: DNAtype_values, dtype: int64
```

In [40]:
```
plt.scatter(y = "DNAtype_values", x = "index", data = dataframe['DNAtype_values'].value_counts().reset_index())
plt.xlabel("Count")
plt.ylabel("DNAtype")
plt.title("Frequency count")
plt.xticks(
    rotation=45, horizontalalignment="right", fontweight="light", fontsize="x-large")
plt.show()
```



In [52]:
```
dataframe_corr = pd.read_csv("codon_usage.csv")
```

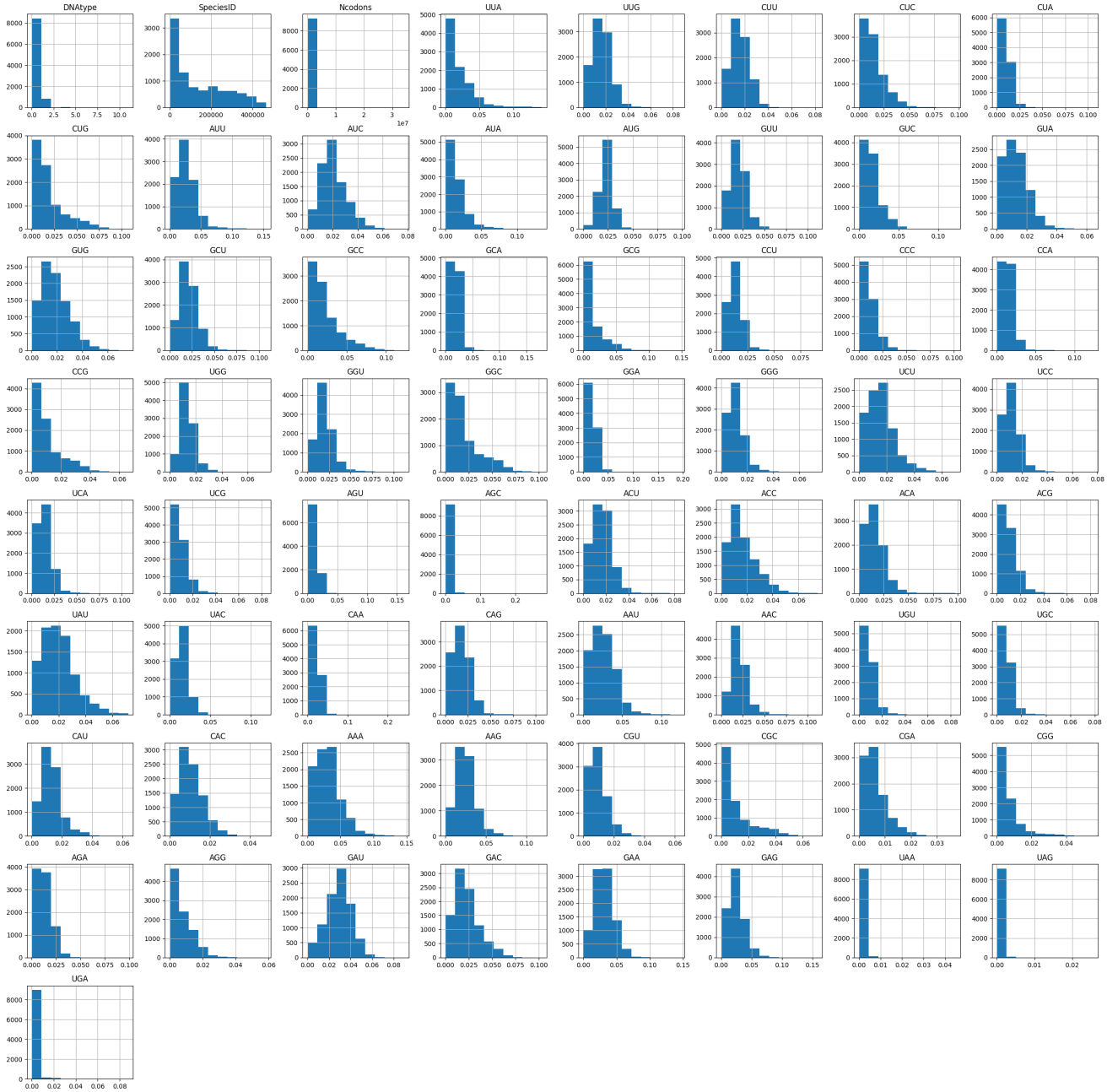
In [53]:
```
dataframe_corr.drop(['Kingdom', 'DNAtype', 'SpeciesID', 'Ncodons', 'SpeciesName'], axis=1)
```

Out[53]:

	UUU	UUC	UUA	UUG	CUU	CUC	CUA	CUG	AUU	AUC	...	CGG	AGA	AGG	GAU	GAC	GAA	GA
0	0.01654	0.01203	0.00050	0.00351	0.01203	0.03208	0.00100	0.04010	0.00551	0.02005	...	0.00451	0.01303	0.03559	0.01003	0.04612	0.01203	0.0436
1	0.02714	0.01357	0.00068	0.00678	0.00407	0.02849	0.00204	0.04410	0.01153	0.02510	...	0.00136	0.01696	0.03596	0.01221	0.04545	0.01560	0.0441
2	0.01974	0.0218	0.01357	0.01543	0.00782	0.01111	0.01028	0.01193	0.02283	0.01604	...	0.00596	0.01974	0.02489	0.03126	0.02036	0.02242	0.0246
3	0.01775	0.02245	0.01619	0.00992	0.01567	0.01358	0.00940	0.01723	0.02402	0.02245	...	0.00366	0.01410	0.01671	0.03760	0.01932	0.03029	0.0344
4	0.02816	0.01371	0.00767	0.03679	0.01380	0.00548	0.00473	0.02076	0.02716	0.00867	...	0.00604	0.01494	0.01734	0.04148	0.02483	0.03359	0.0367
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
9272	0.02802	0.01378	0.01148	0.04088	0.01562	0.00322	0.00505	0.01102	0.03537	0.01516	...	0.00230	0.00827	0.00230	0.03813	0.02297	0.05466	0.0147
9273	0.0162	0.02558	0.00767	0.01364	0.01279	0.00938	0.00853	0.01194	0.01194	0.02302	...	0.00512	0.01535	0.01194	0.02728	0.04263	0.02728	0.0221
9274	0.01281	0.01873	0.00558	0.01478	0.01380	0.01774	0.00657	0.01741	0.01873	0.01938	...	0.00230	0.01117	0.00558	0.03219	0.03121	0.02891	0.0269
9275	0.01429	0.05168	0.00168	0.01429	0.00630	0.00882	0.00420	0.02017	0.01597	0.01429	...	0.00588	0.00756	0.00420	0.02521	0.02353	0.03571	0.0239
9276	0.01176	0.01961	0.00000	0.01457	0.00000	0.01176	0.00000	0.06779	0.03473	0.05546	...	0.00392	0.00000	0.00000	0.02353	0.02353	0.00392	0.0352

9277 rows × 64 columns

In [59]:
```
dataframe_corr.hist(figsize=(30,30))
plt.show()
```



Supervised algorithm - Random forest Classsifier

In [76]:
```
dataframe_cls = pd.read_csv('codon_usage.csv')

dataframe_cls["Kingdom"] = dataframe_cls["Kingdom"].astype('category')
dataframe_cls["Kingdom_category"] = dataframe_cls["Kingdom"].cat.codes
dataframe_cls.head()
```

Out[76]:

	Kingdom	DNAtype	SpeciesID	Ncodons	SpeciesName	UUU	UUC	UUA	UUG	CUU	...	AGA	AGG	GAU	GAC	GAA	GA(
0	vrl	0	100217	1995	Epizootic haematopoietic necrosis virus	0.01654	0.01203	0.00050	0.00351	0.01203	...	0.01303	0.03559	0.01003	0.04612	0.01203	0.0436
1	vrl	0	100220	1474	Bohle iridovirus	0.02714	0.01357	0.00068	0.00678	0.00407	...	0.01696	0.03596	0.01221	0.04545	0.01560	0.0441
2	vrl	0	100755	4862	Sweet potato leaf curl virus	0.01974	0.02180	0.01357	0.01543	0.00782	...	0.01974	0.02489	0.03126	0.02036	0.02242	0.0246
3	vrl	0	100880	1915	Northern cereal mosaic virus	0.01775	0.02245	0.01619	0.00992	0.01567	...	0.01410	0.01671	0.03760	0.01932	0.03029	0.0344
4	vrl	0	100887	22831	Soil-borne cereal mosaic virus	0.02816	0.01371	0.00767	0.03679	0.01380	...	0.01494	0.01734	0.04148	0.02483	0.03359	0.0367

5 rows × 70 columns

In [77]:
```
dataframe_cls1 = dataframe_cls.loc[:,dataframe_cls.columns[6:]]
dataframe_cls1.head()
```

Out[77]:

	UUC	UUA	UUG	CUU	CUC	CUA	CUG	AUU	AUC	AUA	...	AGA	AGG	GAU	GAC	GAA	GAG	UAA
0	0.01203	0.00050	0.00351	0.01203	0.03208	0.00100	0.04010	0.00551	0.02005	0.00752	...	0.01303	0.03559	0.01003	0.04612	0.01203	0.04361	0.00251
1	0.01357	0.00068	0.00678	0.00407	0.02849	0.00204	0.04410	0.01153	0.02510	0.00882	...	0.01696	0.03596	0.01221	0.04545	0.01560	0.04410	0.00271
2	0.02180	0.01357	0.01543	0.00782	0.01111	0.01028	0.01193	0.02283	0.01604	0.01316	...	0.01974	0.02489	0.03126	0.02036	0.02242	0.02468	0.00391
3	0.02245	0.01619	0.00992	0.01567	0.01358	0.00940	0.01723	0.02402	0.02245	0.02507	...	0.01410	0.01671	0.03760	0.01932	0.03029	0.03446	0.00261
4	0.01371	0.00767	0.03679	0.01380	0.00548	0.00473	0.02076	0.02716	0.00867	0.01310	...	0.01494	0.01734	0.04148	0.02483	0.03359	0.03679	0.00000

5 rows × 64 columns

In [78]:
```
X = dataframe_cls1.iloc[:,:-1]
y = dataframe_cls["Kingdom_category"]

# standardize the dataset
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# split into train and test set
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, stratify=y, test_size=0.20, random_state=42)
```

In [79]:
```
classifier = RandomForestClassifier(n_estimators=100)

# Train the model using the training sets
classifier.fit(X_train, y_train)
```

Out[79]:
```
RandomForestClassifier()
```
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 [80]:
```
y_pred = classifier.predict(X_test)
```

In [87]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.77      0.40      0.53        25
           1       0.89      0.96      0.93       584
           2       0.99      0.54      0.70       128
           3       0.95      0.45      0.62        44
           4       0.00      0.00      0.00         4
           5       0.90      0.93      0.92       505
           6       0.91      0.96      0.93       566

    accuracy                           0.90      1856
   macro avg       0.77      0.61      0.66      1856
weighted avg       0.90      0.90      0.90      1856
```

In [81]:
```
print("Accuracy:", accuracy_score(y_test, y_pred))
```

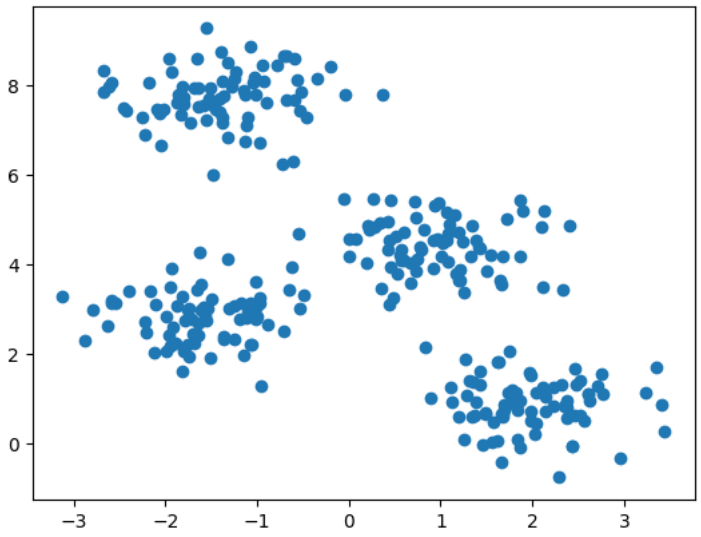
```
Accuracy: 0.9030172413793104
```

In [ ]:
```
```

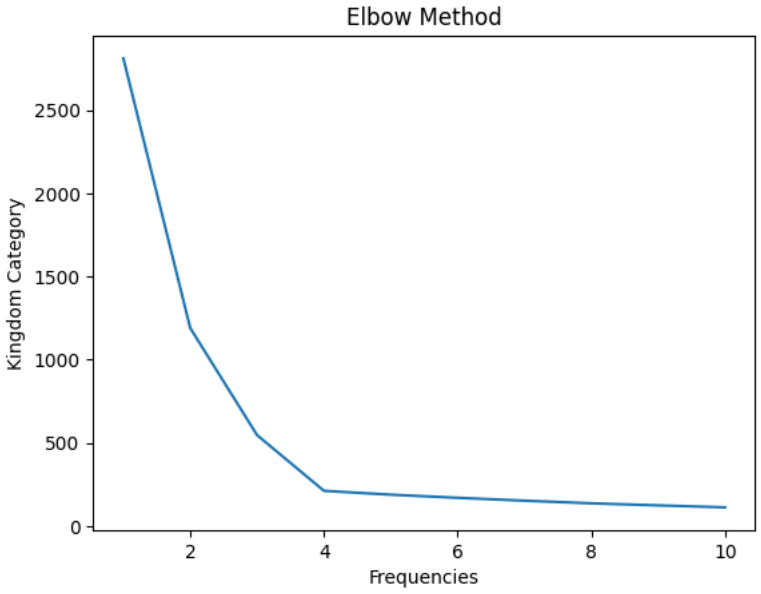
Unsupervised algorithm - K-Means

In [93]:
```
X, y = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)
plt.scatter(X[:,0], X[:,1])
```

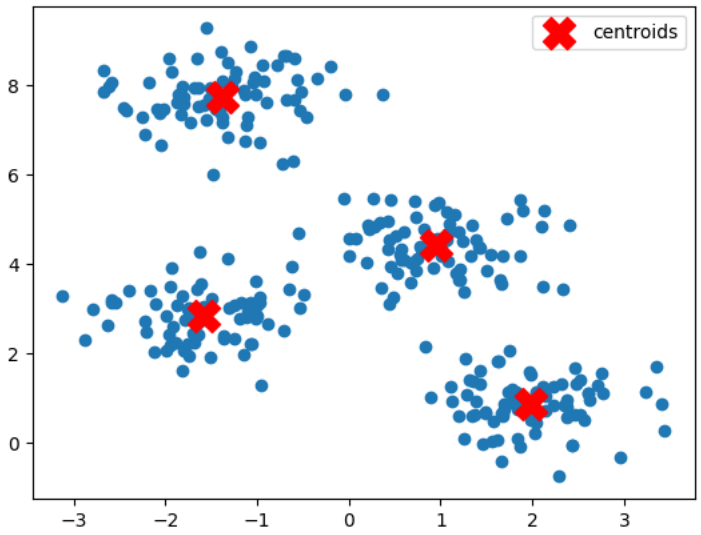
Out[93]:
```
<matplotlib.collections.PathCollection at 0x24384f63c10>
```



In [94]:
```
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Frequencies')
plt.ylabel('Kingdom Category')
plt.show()
```



In [98]:
```
kmeans = KMeans(n_clusters=4, init='k-means++', max_iter=300, n_init=10, random_state=0)
y_pred = kmeans.fit_predict(X)
plt.scatter(X[:,0], X[:,1])
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=300, c="r", marker="X", label="centroids")
plt.legend()
plt.show()
```



In [99]:
```
from sklearn.datasets import load_digits
```

In [101]:
```
dataframe_cls1 = load_digits()
dataframe_cls1.data.shape
```

Out[101]: (1797, 64)

In [103]:
```
kmeans = KMeans(n_clusters=10, random_state=0)
clusters = kmeans.fit_predict(dataframe_cls1.data)
kmeans.cluster_centers_.shape
```

Out[103]: (10, 64)

In [114]:
```
# Project the data:
tsne = TSNE(n_components=2, init='random', random_state=0)
dataframe_cls1_proj = tsne.fit_transform(dataframe_cls1.data)

# Compute the clusters
kmeans = KMeans(n_clusters=10, random_state=0)
clusters = kmeans.fit_predict(dataframe_cls1_proj)
```

In [115]:
```
# Permute the labels
labels = np.zeros_like(clusters)
for i in range(10):
    mask = (clusters == i)
    labels[mask] = mode(dataframe_cls1.target[mask])[0]
```

In [116]:
```
# Compute the accuracy
accuracy_score(dataframe_cls1.target, labels)
```

Out[116]: 0.9426822481914302

PCA is done on another document attached