Principal Component Analysis on Codon Usage dataset

In [2]:
```
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


from sklearn.decomposition import PCA
%matplotlib inline
```

Loading the dataset

In [8]:
```
dataframe = pd.read_csv("codon_usage.csv")
```

Creating Categories for Kindom Attribute

In [9]:
```
dataframe["Kingdom"] = dataframe["Kingdom"].astype('category')
dataframe["Kingdom_category"] = dataframe["Kingdom"].cat.codes
dataframe.head()
```

Out[9]:

	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 [10]:
```
dataframe1 = dataframe.loc[:,dataframe.columns[6:]]
dataframe1.head()
```

Out[10]:

	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 [26]:
```
dataframe1.columns
```

Out[26]:
```
Index(['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', 'Kingdom_category'],
      dtype='object')
```

In [27]:
```
dataframe_columns = ['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']
```

Standardizing ,Training and testing the dataset by choosing 20% of test_size

In [30]:
```
# Separating out the Kingdome_category

x = dataframe1.loc[:,dataframe_columns].values
y = dataframe1.loc[:,['Kingdom_category']].values
```

In [ ]:
```
# 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 [31]:
```
classifier = RandomForestClassifier(n_estimators=100)
classifier.fit(X_train, y_train)
```

Out[31]:
```
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 [32]:
```
from sklearn.preprocessing import StandardScaler
```

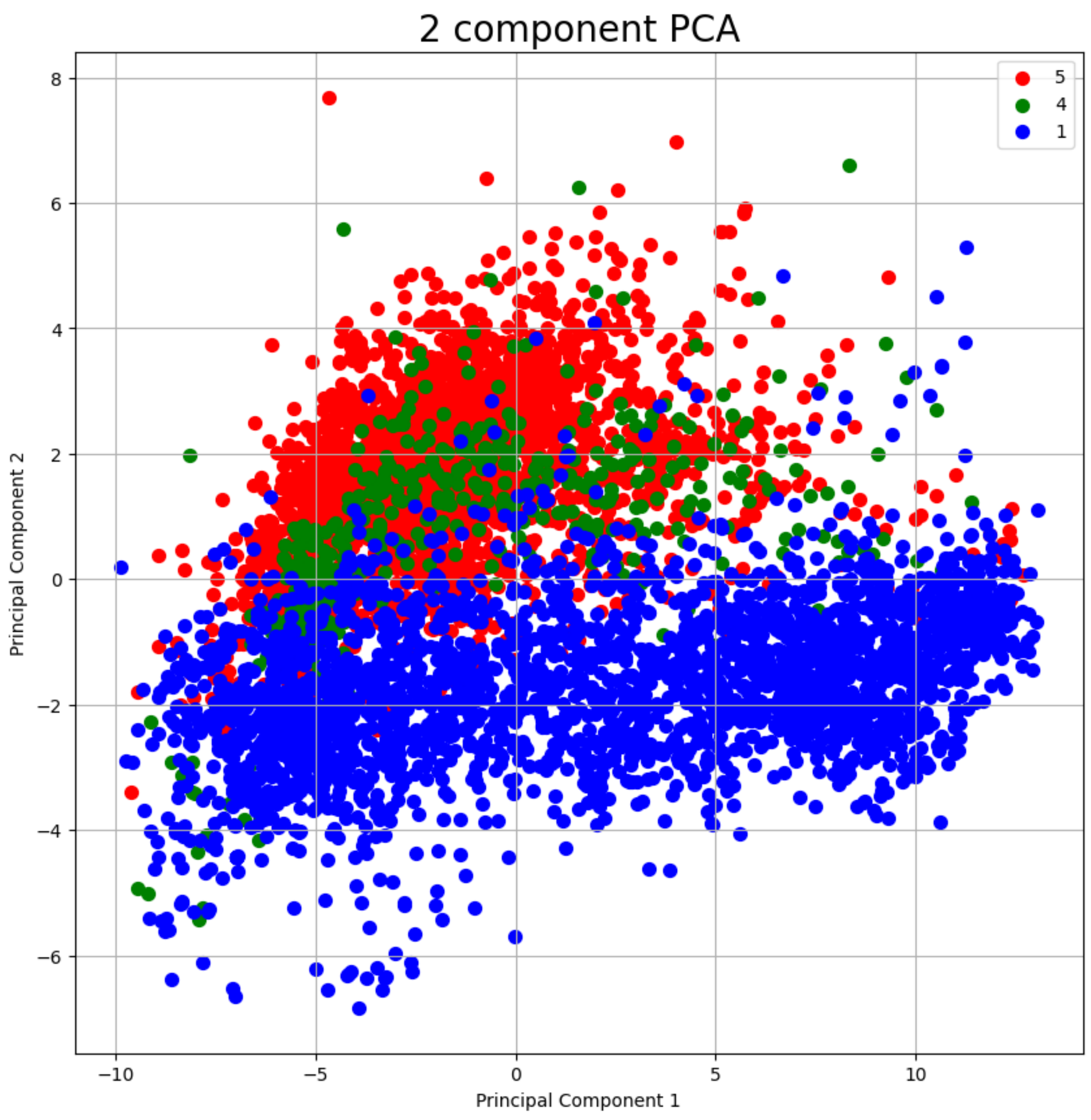
In [33]:
```
X = StandardScaler().fit_transform(X)
```

Principal Component Analysis (PCA)

In [34]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(X)
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['principal component 1', 'principal component 2'])
```

In [35]:
```
finalDf = pd.concat([principalDf, dataframe1[['Kingdom_category']]], axis = 1)
```

In [59]:
```
fig = plt.figure(figsize = (10,10))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 10)
ax.set_ylabel('Principal Component 2', fontsize = 10)
ax.set_title('2 component PCA', fontsize = 20)
targets = [5 , 4 , 1]
color = ['r','b','g']
for Kingdom_category, color in zip(targets,colors):
    indicesToKeep = finalDf['Kingdom_category'] == Kingdom_category
    ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']
               , finalDf.loc[indicesToKeep, 'principal component 2']
               , c = color
               , s = 50)
ax.legend(targets)
ax.grid()
```



variance ratio by PCA

In [60]:
```
pca.explained_variance_ratio_
```

Out[60]:
```
array([0.41841227, 0.07293357])
```

Adding the data and again working on PCA to compare

In [76]:
```
dataframe_added = pd.read_csv("codon_usage.csv")
```

In [77]:
```
dataframe_added["Kingdom"] = dataframe_added["Kingdom"].astype('category')
dataframe_added["Kingdom_category"] = dataframe_added["Kingdom"].cat.codes
dataframe_added.head()
```

Out[77]:

	Kingdom	DNAtype	SpeciesID	Ncodons	SpeciesName	UUU	UUC	UUA	UUG	CUU	...	AGA	AGG	GAU	GAC	GAA	GA
0	vrl	0.0	100217.0	1995.0	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	0.0	100220.0	1474.0	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	0.0	100755.0	4862.0	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	0.0	100880.0	1915.0	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	0.0	100887.0	22831.0	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 [91]:
```
dataframe_added1 = dataframe_added.loc[:,dataframe.columns[6:]]
dataframe_added1.head()
```

Out[91]:

	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 [79]:
```
dataframe_columns1 = ['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']
```

In [92]:
```
def clean_dataset(dataframe_added1):
    assert isinstance(dataframe_added1, pd.DataFrame), "df needs to be a pd.DataFrame"
    dataframe_added1.dropna(inplace=True)
    indices_to_keep = ~dataframe_added1.isin([np.nan, np.inf, -np.inf]).any(1)
    return dataframe_added1[indices_to_keep].astype(np.float64)
```

In [93]:
```
X = dataframe_added1.loc[:,dataframe_columns1].values
# Separating out the target
Y = dataframe_added1.loc[:,['Kingdom_category']].values
```

In [94]:
```

# 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=0.30, random_state=42)
```

In [98]:
```
classifier = RandomForestClassifier(n_estimators=100)
classifier.fit(X_train, _train)
```

Out[98]:
```
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.