Bird Personality Data Analysis Using Clustering and PCA

Clustering Plots:

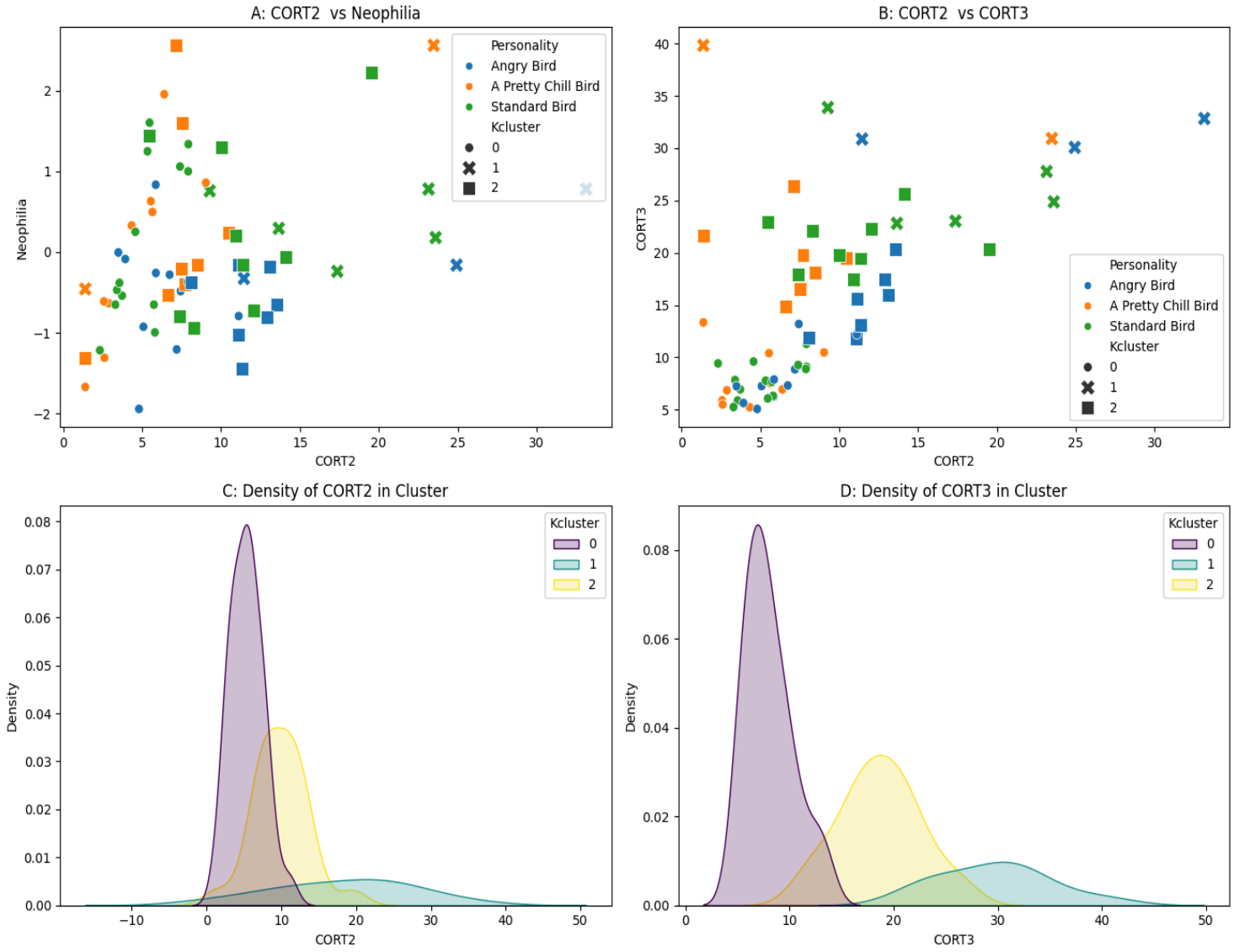

Figure 1. K-means cluster analysis for “Angry birds(Cardinal Personalities)” dataset of a 3-group cluster analysis. Panel A shows CORT2 and Neophilia have very little separation that corresponds to the clusters. Panel B shows the clusters on CORT2 and CORT3, which have much better separation - especially along the x axis, showing that the cluster analysis is using CORT2 pretty extensively. Panel C and Panel D show kernel density estimates of CORT2 and CORT3 , with the three clusters more clearly separated for CORT3(Panel D).

PCA plots

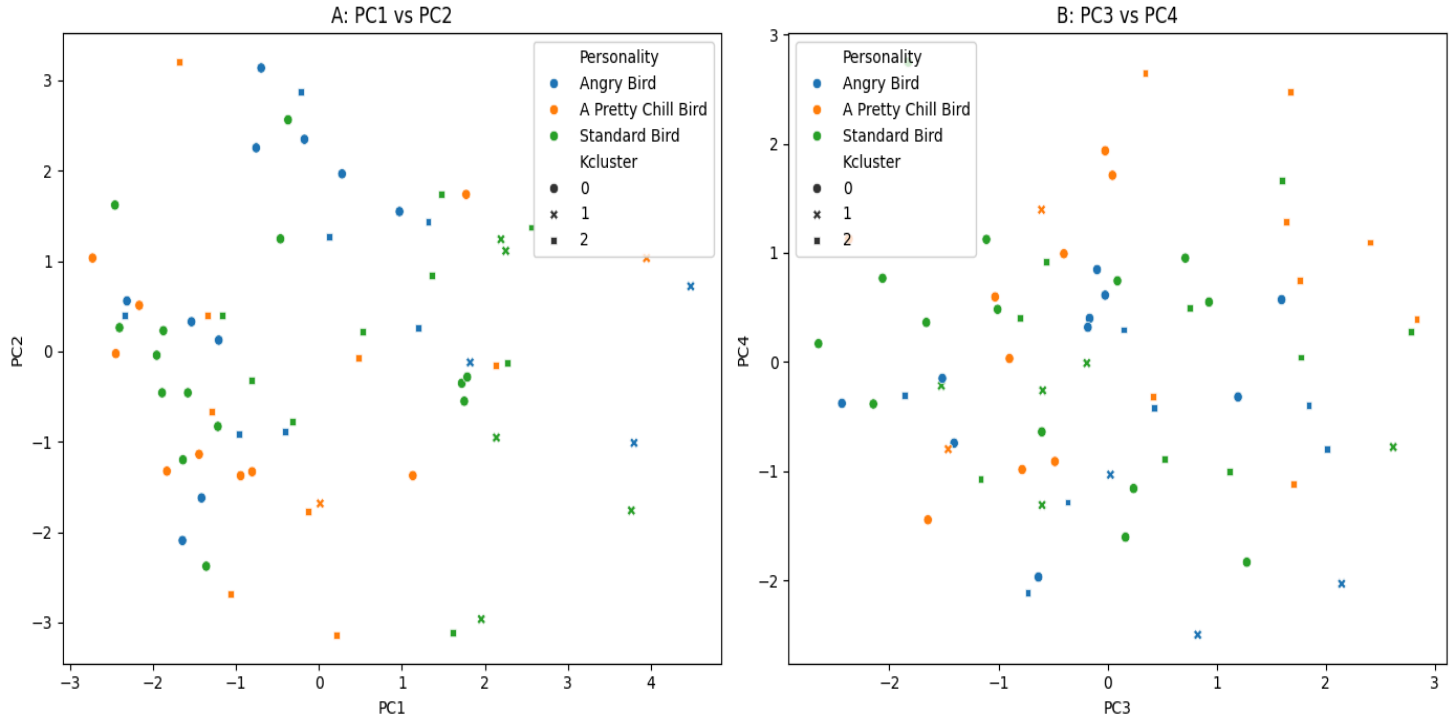

Figure 2. Cluster and true personalities assignment displayed on two principal component plots. While Panel A shows principal components analysis for PC1 to PC2 does not align well with the cluster assignments, there is no separation in PC3 to PC4, where personalities and clusters overlap in Panel B.

Questions:

1) Run a cluster analysis with 3 groups. Do they match well with the personality group?
2) What variables seem the most useful for separating out the clusters with the personality column?
3) Does a PCA seem to line up with either the grouping variable or the clusters?

Interpretation Results answering all the three questions:

Clusters did not particularly match the personality group here, nor did they work well within the principal components analysis. Kindly find the screenshot below that highlights significance of each variable to the respective principal components.
The most important contribution to PC1 was “CORT2”. For PC2, the most important feature is “WingL”. For PC3, its “Month” that contributes the most followed by “Boldness” and for PC4, the significant feature is “Aggression”

	PC1	PC2	PC3	PC4
Month	0.107077	-0.085743	-0.478157	-0.293012
Habitat	-0.399288	-0.368258	0.085276	-0.005933
Location	-0.401507	-0.276135	0.307073	0.094402
Sex	-0.107104	0.446608	0.078511	0.223454
Weight	-0.158150	0.281348	0.179909	-0.182971
WingL	-0.068115	0.492334	0.134789	0.178811
TarsusL	0.002499	0.250780	0.331035	0.030458
CORT1	0.369636	-0.211264	0.223772	0.149910
CORT2	0.404442	-0.128189	0.120265	-0.132070
CORT3	0.345017	-0.083681	0.205620	-0.153266
Exploration	-0.071772	-0.179987	0.304668	-0.181427
Neophobia	0.285735	0.105277	0.255728	-0.301858
Neophilia	0.345143	-0.003881	0.033816	0.512210
Aggression	-0.025995	0.289523	-0.116025	-0.524894
Boldness	0.030500	0.038728	-0.473155	0.252136

CORT2 and CORT3 visually appear to be more correlated with the clusters than CORT2 and Neophilia features.
Panel C and Panel D in Figure 1. show kernel density estimates of CORT2 and CORT3, with the three clusters more clearly separated for CORT3(Panel D).

Code snippet and steps to answer the questions:

In order to answer the 3 questions, I had to perform data preprocessing steps like data cleaning, data manipulation and finally perform data visualization to understand any pattern or trend.

Step 1: Perform data preprocessing and data cleaning of the Angry birds(Cardinal Personality) dataset.
Step 2: Convert all categorical objects to numeric to make the dataset ready for K-means clustering.
Step 3: Perform K-means clustering with K=3 as there are 3 personality grouping variables that need to be compared.
Step 4: Perform visualization on the result obtained after clustering.
Step 5: Normalize the dataset to perform Principal component analysis(PCA).
Step 6: Finally, develop an appropriate plot for data visualization.
Step 7: Add interpretation results.

Code snippet for **Cluster plots:**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import copy
import warnings
warnings.filterwarnings('ignore')
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from scipy.stats import zscore

# Define month mapping
month_mapping = {
    'Jan': 1, 'Feb': 2, 'Mar': 3, 'Apr': 4, 'May': 5, 'Jun': 6,
    'Jul': 7, 'Aug': 8, 'Sept': 9, 'Oct': 10, 'Nov': 11, 'Dec': 12
}

# Apply month mapping to the 'Month' column
df['Month'] = df['Month'].map(month_mapping)
# Convert categorical variables to numeric
for feature in df.columns: # Loop through all columns in the dataframe
    if df[feature].dtype == 'object': # Only apply for columns with
categorical strings
```

```
        df[feature] = pd.Categorical(df[feature]).codes # Replace strings
with an integer
# Perform K-means clustering
# Define the model
kmeans = KMeans(n_clusters=3, random_state=0)

# Fit and predict
df_personality['Kcluster'] = kmeans.fit_predict(df)
fig, axes = plt.subplots(2, 2, figsize=(14, 10))

# Plot 1
sns.scatterplot(data=df_personality, y='Neophilia', x='CORT2',
hue='Personality', style='Kcluster', size='Kcluster',sizes=(50,200),
ax=axes[0, 0])
axes[0, 0].set_title('A: CORT2  vs Neophilia')

# Plot 2
sns.scatterplot(data=df_personality, x='CORT2', y='CORT3',
hue='Personality', style='Kcluster', size='Kcluster',sizes=(50,200),
ax=axes[0, 1])
axes[0, 1].set_title('B: CORT2  vs CORT3')

# Plot 3
sns.kdeplot(data=df_personality, x='CORT2', hue='Kcluster', fill=True,
ax=axes[1, 0], palette ='viridis')
axes[1, 0].set_title('C: Density of CORT2 in Cluster')

# Plot 4
sns.kdeplot(data=df_personality, x='CORT3', hue='Kcluster', fill=True,
ax=axes[1, 1], palette ='viridis')
axes[1, 1].set_title('D: Density of CORT3 in Cluster')

plt.tight_layout()
plt.show()
```

Code snippet for **PCA plots:**

```
scaler = StandardScaler()
birds_scaled = scaler.fit_transform(df)
pca = PCA(n_components=4)
pca_result = pca.fit_transform(birds_scaled)
```

```
# Add to DataFrame
PCA_output = pd.DataFrame(data = pca_result, columns = ['PC'+str(i) for i
in range(1, 5)])
PCA_output['Kcluster'] = df_personality['Kcluster'].values
PCA_output['Personality'] = df_personality['Personality'].values

# Visualize
fig, axes = plt.subplots(1, 2, figsize=(14, 6))

sns.scatterplot(data=PCA_output, x='PC1', y='PC2', hue='Personality',
style='Kcluster', ax=axes[0])
axes[0].set_title('A: PC1 vs PC2')

sns.scatterplot(data=PCA_output, x='PC3', y='PC4', hue='Personality',
style='Kcluster', ax=axes[1])
axes[1].set_title('B: PC3 vs PC4')

plt.tight_layout()
plt.show()

# Standard deviation of each component
std_dev = np.sqrt(pca.explained_variance_)

# Variance of each component
variance = pca.explained_variance_

# Percentage of variance explained by each component
variance_ratio = pca.explained_variance_ratio_

# Convert to DataFrame for clearer visualization
df_pca = pd.DataFrame({'stdev': std_dev,
                       'variance': variance,
                       '% of variance': variance_ratio},
                       index=[f'PC{i+1}' for i in
range(std_dev.shape[0])])

# Convert "Percentage of Variance" to a percentage
```

```
df_pca['% of variance'] = df_pca['% of variance'].map(lambda x:
round(x*100, 2))

df_pca
# Create a dataframe of the PCA components
df_components = pd.DataFrame(pca.components_, columns=df.columns)

# Transpose it, so that variables are the index and PCs are the columns
df_components = df_components.T

# Give the columns meaningful names
df_components.columns = [f'PC{i+1}' for i in
range(df_components.shape[1])]

# Show the dataframe
df_components
```