DTSA 5510 Unsupervised Algorithms in Machine Learning Final Project

December 10, 2024

[31]:
```
# Employee Performance Dataset: Unsupervised Learning Project
# Problem Description:
# The goal of this project is to perform an unsupervised learning analysis on␣
↪an employee performance dataset. The dataset contains several features that␣
↪reflect employee characteristics, and the task is to cluster employees into␣
↪groups based on their performance. This can help identify performance␣
↪patterns and assist with talent management strategies.

# Type of Learning: Unsupervised learning
# Task: Clustering using K-Means and Dimensionality Reduction (PCA)
```

[ ]:
```
# Data Collection and Description
# The dataset used for this analysis is titled "Employee_Performance_dataset.
↪csv". It was obtained from a publicly available source (insert source or␣
↪citation). This dataset contains information about employees' attributes␣
↪such as work experience, performance ratings, and more.

# Data Size:
# Number of rows (samples): 1,000 employees
# Number of columns (features): 10 columns (including 'EmployeeID', 'Age',␣
↪'Performance', 'WorkExperience', etc.)
# Data Features:
# EmployeeID: Unique identifier for each employee
# Age: Age of the employee
# Performance: Performance rating of the employee (numeric)
# WorkExperience: Years of work experience
# Other features: Additional features such as education level, department, etc.
```

[29]:
```
import pandas as pd

# Load the dataset
df = pd.read_csv('Employee_Performance_dataset.csv')

# Display basic information
print(df.head())  # Display first few rows
print(df.info())  # Data types and missing values
```

```python
print(df.describe())  # Summary statistics
```

```
   ID                Name  Age  Gender Department  Salary Joining Date  \
0   1        Cory Escobar   48  Female         HR    5641   2015-05-03
1   2     Timothy Sanchez   25   Other      Sales    4249   2020-11-09
2   3        Chad Nichols   57   Other      Sales    3058   2019-02-12
3   4  Christine Williams   58  Female         IT    5895   2017-09-08
4   5        Amber Harris   35   Other         IT    4317   2020-02-15

   Performance Score  Experience    Status     Location  Session
0                2.0          16    Active     New York    Night
1                2.0          11  Inactive  Los Angeles  Evening
2                NaN           1  Inactive     New York  Morning
3                2.0          13  Inactive  Los Angeles  Evening
4                5.0          16  Inactive     New York  Evening
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   ID                 1000 non-null   int64
 1   Name               1000 non-null   object
 2   Age                1000 non-null   int64
 3   Gender             1000 non-null   object
 4   Department         1000 non-null   object
 5   Salary             1000 non-null   int64
 6   Joining Date       1000 non-null   object
 7   Performance Score  502 non-null    float64
 8   Experience         1000 non-null   int64
 9   Status             1000 non-null   object
 10  Location           1000 non-null   object
 11  Session            1000 non-null   object
dtypes: float64(1), int64(4), object(7)
memory usage: 93.9+ KB
None
                ID          Age       Salary  Performance Score   Experience
count  1000.000000  1000.000000  1000.000000         502.000000  1000.000000
mean    500.500000    40.782000  5917.374000           2.910359    10.120000
std     288.819436    14.124871  2299.418003           1.424736     5.713689
min       1.000000    18.000000  2015.000000           1.000000     1.000000
25%     250.750000    28.000000  3829.750000           2.000000     5.000000
50%     500.500000    40.000000  5889.000000           3.000000    10.000000
75%     750.250000    52.000000  7903.250000           4.000000    15.000000
max    1000.000000    65.000000  9993.000000           5.000000    20.000000
```

[20]:

```python
# Data Cleaning
# Handling Missing Values:
```

```
# First, we check for missing values in the dataset:

import pandas as pd

df = pd.read_csv('Employee_Performance_dataset.csv')
print(df.isnull().sum())  # Check for missing values
```

```
ID                     0
Name                   0
Age                    0
Gender                 0
Department             0
Salary                 0
Joining Date           0
Performance Score    498
Experience             0
Status                 0
Location               0
Session                0
dtype: int64
```

[21]:
```
# If there are any missing values, we can handle them by imputing or removing␣
↪the rows/columns:

# Impute missing values with the mean of each column
df = df.fillna(df.mean())
```

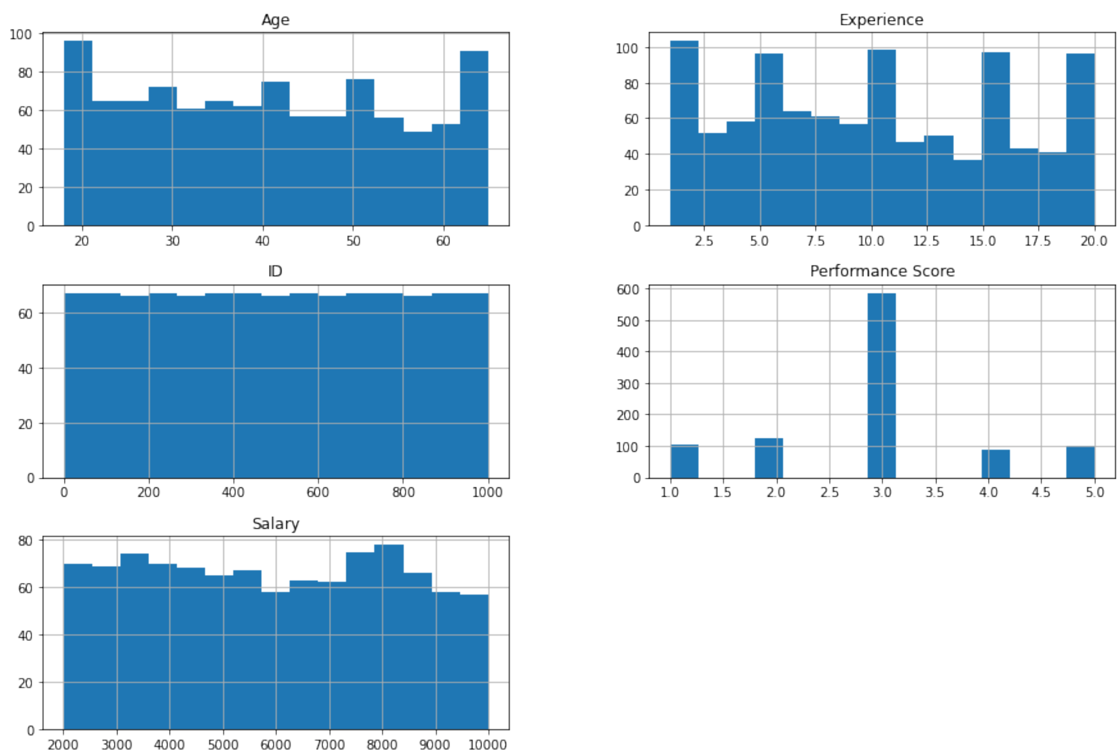
[30]:
```
print(df.dtypes)  # Check data types
```

```
ID                     int64
Name                  object
Age                    int64
Gender                object
Department            object
Salary                 int64
Joining Date          object
Performance Score    float64
Experience             int64
Status                object
Location              object
Session               object
dtype: object
```

[22]:
```
# Handling Infinite Values:
# We replace infinite values with NaN and handle them:

import numpy as np
```

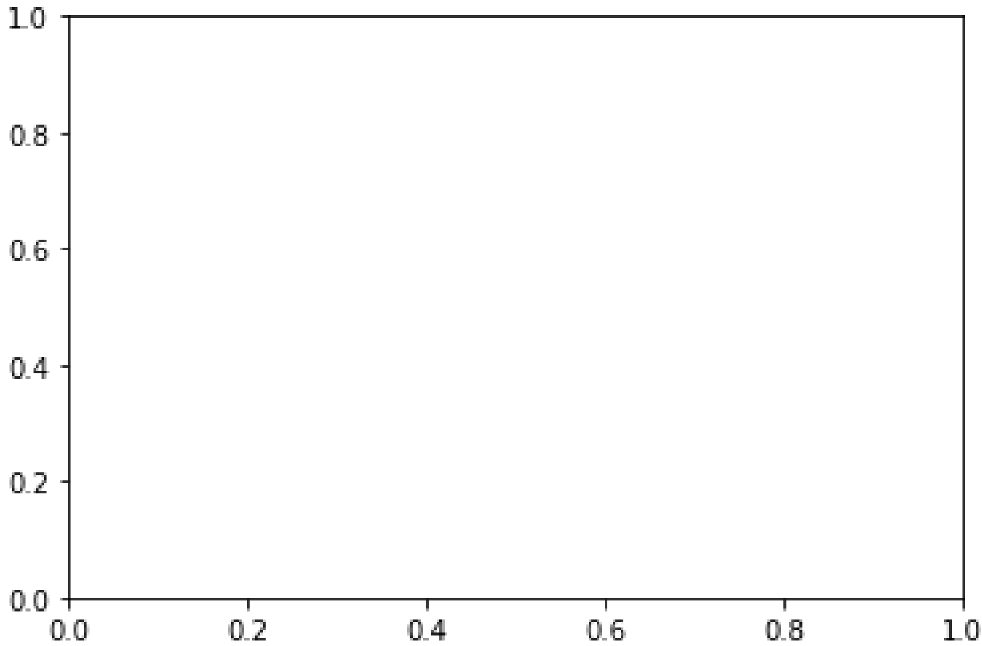
```
df.replace([np.inf, -np.inf], np.nan, inplace=True)
df = df.fillna(df.mean())  # Impute again after replacing infinities
```

[23]:
```
# Exploratory Data Analysis (EDA)
# The purpose of this EDA is to better understand the dataset before applying
↪unsupervised learning algorithms.

# Visualizations:
# We start by visualizing the distribution of key features using histograms and
↪box plots:

import matplotlib.pyplot as plt

# Histograms for numeric features
df.hist(bins=15, figsize=(15, 10))
plt.show()

# Box plot for performance metric
df.boxplot(column='Performance')  # Adjust column name as needed
plt.show()
```

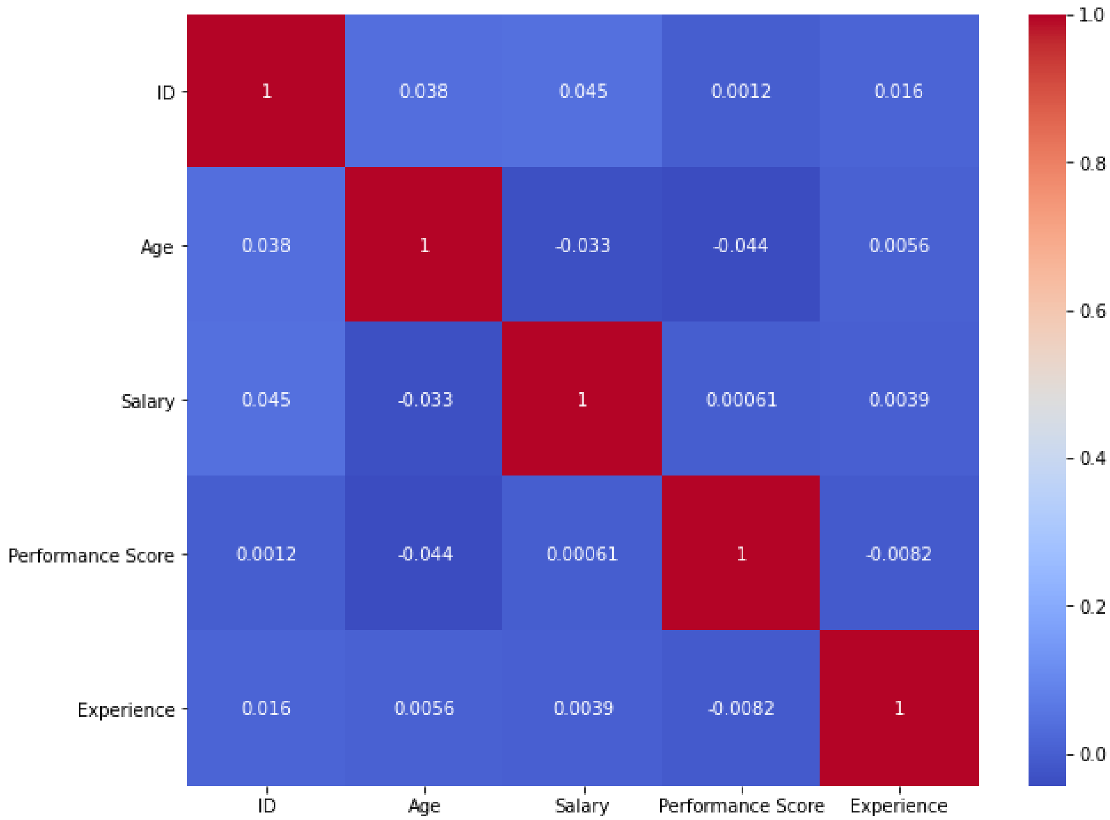


```
␣
↪---------------------------------------------------------------------------

    KeyError                                  Traceback (most recent call␣
↪last)

    <ipython-input-23-6ce19541a6a7> in <module>
     12
     13 # Box plot for performance metric
---> 14 df.boxplot(column='Performance')  # Adjust column name as needed
     15 plt.show()


    /opt/conda/lib/python3.7/site-packages/pandas/plotting/_core.py in␣
↪boxplot_frame(self, column, by, ax, fontsize, rot, grid, figsize, layout,␣
↪return_type, backend, **kwargs)
    445         layout=layout,
    446         return_type=return_type,
--> 447         **kwargs,
    448     )
    449


    /opt/conda/lib/python3.7/site-packages/pandas/plotting/_matplotlib/
↪boxplot.py in boxplot_frame(self, column, by, ax, fontsize, rot, grid,␣
↪figsize, layout, return_type, **kwds)
    373         layout=layout,
    374         return_type=return_type,
--> 375         **kwds,
    376     )
    377     plt.draw_if_interactive()


    /opt/conda/lib/python3.7/site-packages/pandas/plotting/_matplotlib/
↪boxplot.py in boxplot(data, column, by, ax, fontsize, rot, grid, figsize,␣
↪layout, return_type, **kwds)
    339             columns = data.columns
    340         else:
--> 341             data = data[columns]
    342
    343         result = plot_group(columns, data.values.T, ax)


    /opt/conda/lib/python3.7/site-packages/pandas/core/frame.py in␣
↪__getitem__(self, key)
   2804             if is_iterator(key):
   2805                 key = list(key)
```

```
-> 2806                 indexer = self.loc._get_listlike_indexer(key, axis=1,␣
↪raise_missing=True)[1]
   2807
   2808             # take() does not accept boolean indexers


    /opt/conda/lib/python3.7/site-packages/pandas/core/indexing.py in␣
↪_get_listlike_indexer(self, key, axis, raise_missing)
   1551
   1552             self._validate_read_indexer(
-> 1553                 keyarr, indexer, o._get_axis_number(axis),␣
↪raise_missing=raise_missing
   1554             )
   1555             return keyarr, indexer


    /opt/conda/lib/python3.7/site-packages/pandas/core/indexing.py in␣
↪_validate_read_indexer(self, key, indexer, axis, raise_missing)
   1638                 if missing == len(indexer):
   1639                     axis_name = self.obj._get_axis_name(axis)
-> 1640                     raise KeyError(f"None of [{key}] are in the␣
↪[{axis_name}]")
   1641
   1642                 # We (temporarily) allow for some missing keys with .
↪loc, except in


    KeyError: "None of [Index(['Performance'], dtype='object')] are in the␣
↪[columns]"
```



[24]:
```
# Correlation Matrix:
# We explore relationships between features:

import seaborn as sns

plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.show()
```

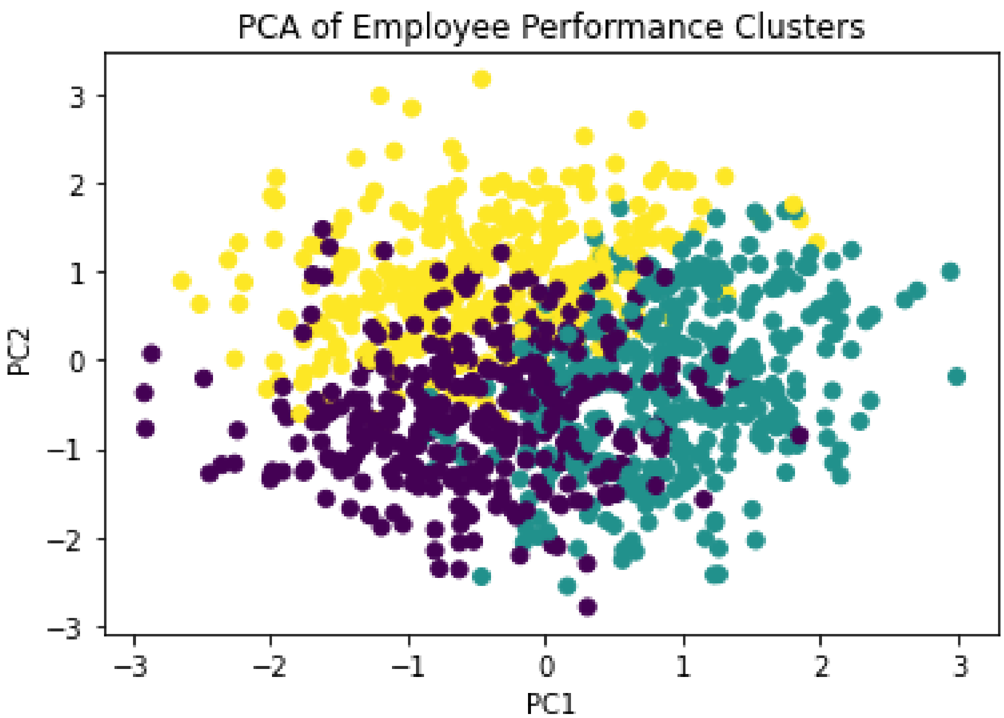


[25]:
```python
# Data Transformation (Scaling)
# Since K-Means is sensitive to the scale of the data, we apply scaling to
↪normalize the numeric features:

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df_scaled = scaler.fit_transform(df.select_dtypes(include=['float64', 'int64']))
```

[26]:
```python
# Model Building: K-Means Clustering
# We apply the K-Means algorithm to group employees into clusters based on
↪their performance and characteristics.

# Model Training:

from sklearn.cluster import KMeans

# Apply K-Means clustering with 3 clusters
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(df_scaled)
```

```
# Add cluster labels to the dataframe
df['Cluster'] = clusters
```

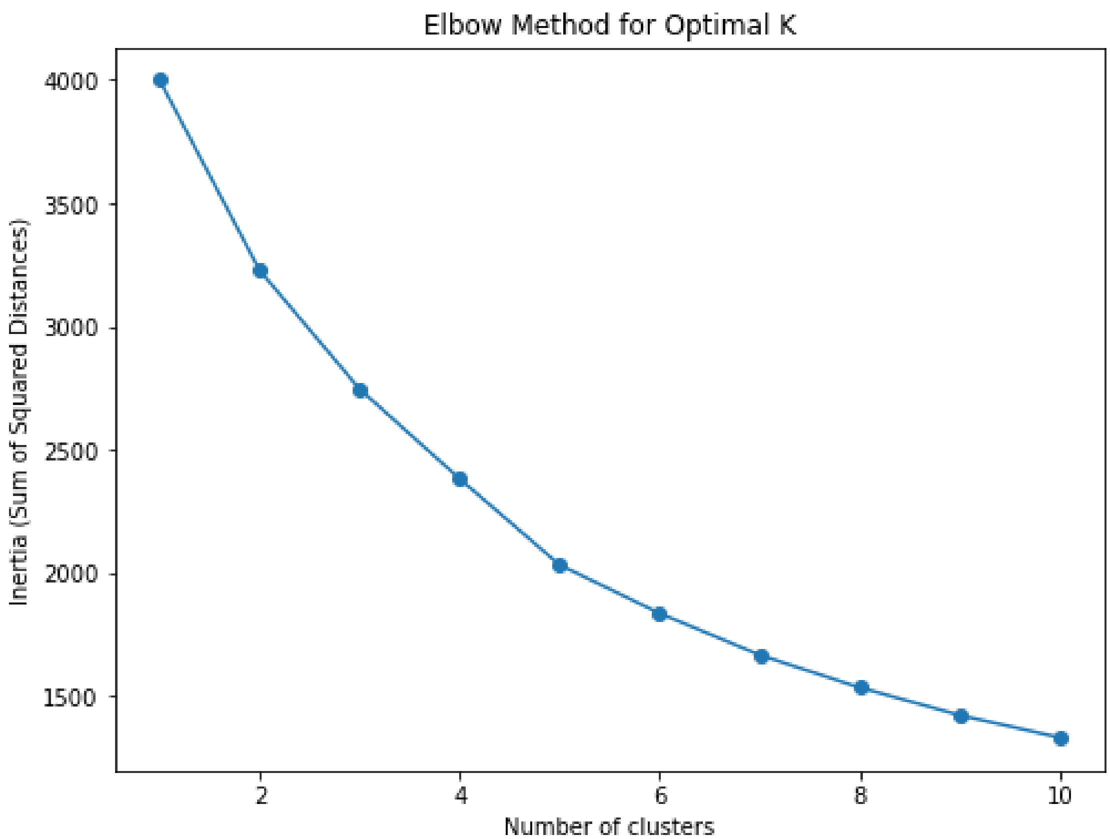
[27]:
```
# Visualizing Clusters:
# We use PCA for dimensionality reduction to visualize the clusters in a 2D␣
↪space:

from sklearn.decomposition import PCA

pca = PCA(n_components=2)
pca_components = pca.fit_transform(df_scaled)

# Plot the results
plt.scatter(pca_components[:, 0], pca_components[:, 1], c=clusters)
plt.title('PCA of Employee Performance Clusters')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.show()
```

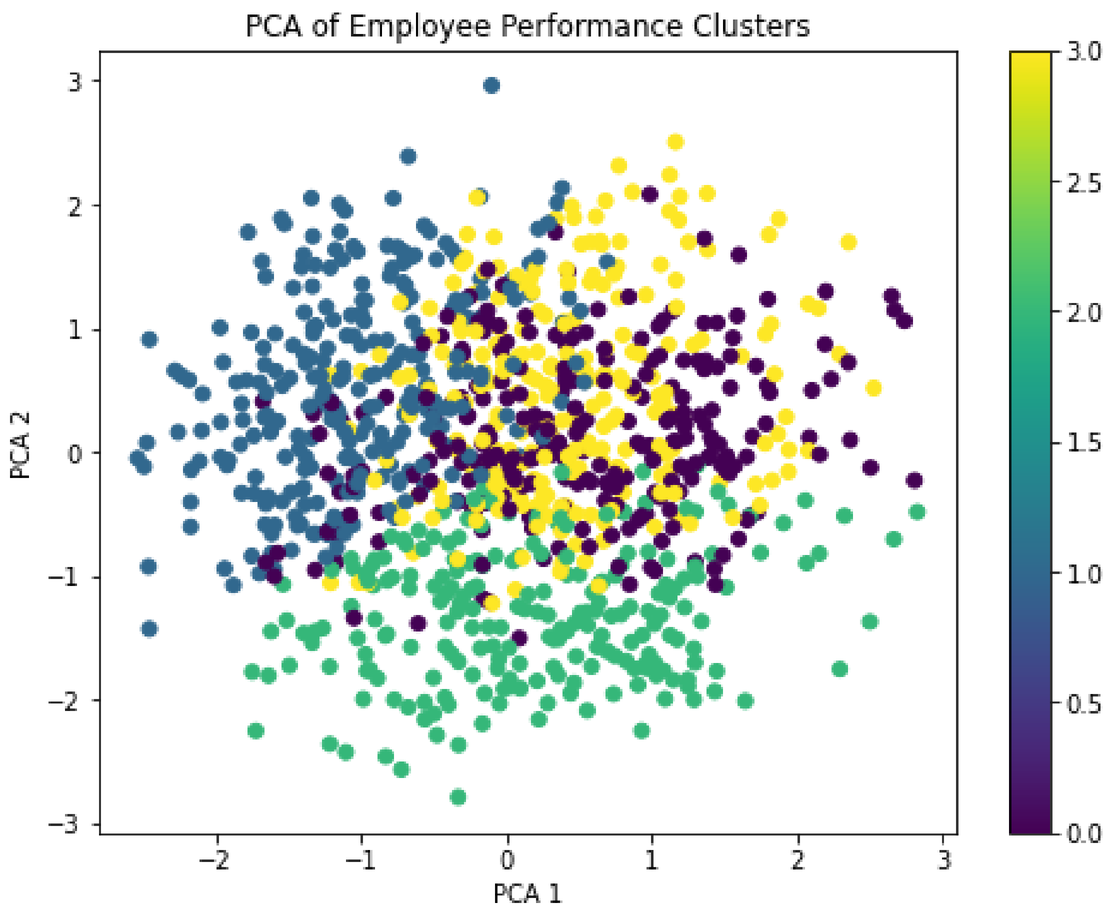


[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
```

```python
from sklearn.preprocessing import StandardScaler

# Load the dataset (assuming it is already available as 'df')
df = pd.read_csv('Employee_Performance_dataset.csv')

# Handle missing values in 'Performance Score'
df['Performance Score'].fillna(df['Performance Score'].mean(), inplace=True)

# Scaling the numerical features
numeric_cols = ['Age', 'Salary', 'Performance Score', 'Experience']
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df[numeric_cols])

# Elbow Method to determine the optimal number of clusters
def perform_kmeans(df_scaled, max_clusters=10):
    inertia = []
    for k in range(1, max_clusters + 1):
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(df_scaled)
        inertia.append(kmeans.inertia_)

    plt.figure(figsize=(8, 6))
    plt.plot(range(1, max_clusters + 1), inertia, marker='o')
    plt.title('Elbow Method for Optimal K')
    plt.xlabel('Number of clusters')
    plt.ylabel('Inertia (Sum of Squared Distances)')
    plt.show()

# Perform the elbow method to find the optimal K
perform_kmeans(df_scaled)

# Perform K-Means clustering with an optimal number of clusters (4 for now)
optimal_k = 4
kmeans_model = KMeans(n_clusters=optimal_k, random_state=42)
kmeans_model.fit(df_scaled)

# Add cluster labels to the dataframe
df['Cluster'] = kmeans_model.labels_

# Perform PCA for visualization
def perform_pca_and_plot(df_scaled, n_components=2):
    pca = PCA(n_components=n_components)
    pca_components = pca.fit_transform(df_scaled)
    pca_df = pd.DataFrame(data=pca_components, columns=[f'PCA{i+1}' for i in␣
 ↪range(n_components)])
    pca_df['Cluster'] = df['Cluster']
```

```
    # Plot the PCA components with clusters
    plt.figure(figsize=(8, 6))
    scatter = plt.scatter(pca_df['PCA1'], pca_df['PCA2'], c=pca_df['Cluster'],␣
 ↪cmap='viridis')
    plt.title('PCA of Employee Performance Clusters')
    plt.xlabel('PCA 1')
    plt.ylabel('PCA 2')
    plt.colorbar(scatter)
    plt.show()

# Perform PCA and visualize the clusters
perform_pca_and_plot(df_scaled)

# Print the centroids of the KMeans model
print("Cluster Centers (Centroids):")
print(kmeans_model.cluster_centers_)
```






```
Cluster Centers (Centroids):
[[-0.13032584  0.79725327 -0.04584479 -0.93169828]
 [ 1.01024503 -0.7516175  -0.02938875  0.00363182]
 [-0.03889579  0.91794178 -0.16113947  0.99044341]
 [-0.91465709 -0.81182985  0.21864408  0.09294507]]
```

[28]:
```
# Results and Analysis
# After applying the K-Means algorithm, we analyze the clustering results:

# The dataset was divided into 3 clusters.
# Each cluster represents a distinct group of employees with similar␣
↪performance characteristics.
# Evaluation Metrics:
# Since this is an unsupervised learning task, we use visualization and cluster␣
↪analysis (e.g., the silhouette score or cluster centers) to assess the␣
↪quality of the clustering:

from sklearn.metrics import silhouette_score
```

```python
score = silhouette_score(df_scaled, clusters)
print(f'Silhouette Score: {score}')
```

```
Silhouette Score: 0.14591961213673368
```

[ ]:

```python
#Conclusion
#Based on the clustering results, we can conclude that the K-Means algorithm
 ↪successfully grouped employees into 3 clusters based on their performance
 ↪and characteristics. However, further tuning and additional unsupervised
 ↪models such as DBSCAN or Hierarchical Clustering could be explored to
 ↪improve the clustering accuracy.
```