Unsupervised Learning:

HR Employee Clustering Analysis

1. Introduction

This analysis explores an HR dataset containing employee demographic and job-related information such as gender, department, job title, location, hire dates, and employment status. The dataset, sourced from a publicly available Human Resources repository, includes over 22,000 valid employee records after cleaning.

The primary objective of this study was to apply unsupervised learning techniques to uncover hidden patterns among employees and group them into meaningful clusters. By performing clustering and dimensionality reduction, the analysis aims to help HR teams identify employee segments that can inform decision-making in retention, training, workforce planning, and engagement strategies.

2. Dataset Source and Description

The dataset used for this analysis was obtained from Figshare, a public research data repository that provides open access to academic datasets. The specific file, titled "Human Resources.csv," contains anonymized employee data designed for educational and analytical use in workforce analytics.

https://figshare.com/articles/dataset/Human_Resources_csv/28780886

The dataset contains approximately 37,000 employee entries, of which about 22,000 are complete and valid for modeling. It includes 13 columns describing demographic and employment information, such as ID, gender, race, department, job title, location (city and state), hire date, and term date. During preprocessing, additional numerical features — Age and Years of Service — were engineered from date fields to represent experience and tenure. The dataset's mix of categorical and temporal features makes it well suited for clustering and dimensionality reduction to reveal underlying workforce patterns.

3. Methodology

Data Preprocessing

Missing values were handled using median imputation for date fields and mode replacement for categorical variables. Irrelevant identifiers (e.g., names, employee IDs) were dropped to maintain anonymity and reduce noise. Categorical features such as department, job title, gender, race, and location were encoded using one-hot encoding

to enable numerical computation. All numerical variables were standardized using StandardScaler to ensure equal weight during clustering.

Clustering and Dimensionality Reduction

The K-Means algorithm was implemented to identify employee clusters based on shared characteristics. The Elbow Method and Silhouette Score were applied to determine the optimal number of clusters. Principal Component Analysis (PCA) reduced dimensionality and allowed for two-dimensional visualization of the cluster structures.

4. Results and Interpretation

The Elbow Method displayed a clear bend around k = 4, suggesting four natural clusters. Silhouette analysis produced scores of k=2 (0.456), k=3 (0.373), k=4 (0.322), k=5 (0.333), and k=6 (0.341). Although the highest score occurred at k = 2, the four-cluster solution (k = 4) was selected for its richer segmentation and better interpretability.

PCA visualization confirmed moderate separation among clusters. The groups likely represent:

New or early-career hires – low age and tenure

Mid-career employees – moderate age and years of service

Long-tenured, senior staff – high experience

Remote or specialized employees – distinctive locations or job roles

These segments reflect real organizational structures where roles overlap but exhibit distinct characteristics useful for HR analysis.

5. Model Evaluation

The final model's Silhouette Score of 0.322 indicates moderate cohesion and separation, common in HR datasets where workforce traits often overlap. The results confirm that the four-cluster model balances statistical validity with business interpretability. It provides actionable segmentation for HR teams to understand workforce composition and plan targeted initiatives such as professional development, retention programs, and onboarding support.

6. Challenges Encountered

Several challenges were encountered during the analysis. The dataset contained missing and inconsistent date values, which required median imputation to maintain data completeness. Managing categorical variables with many unique levels, such as job titles and cities, increased dimensionality after one-hot encoding, making model computation more complex. In addition, interpreting moderate Silhouette Scores was challenging

because employee attributes naturally overlap across departments and tenure levels, limiting perfect cluster separation. Despite these obstacles, careful preprocessing, feature engineering, and model evaluation ensured that the final clustering results remained both statistically reliable and practically interpretable.

7. Deployment and Monitoring Strategy

For deployment, the clustering model can be integrated into an HR analytics dashboard that updates automatically as new employee records are added. Employees can be assigned to clusters in real time to monitor workforce changes and identify emerging patterns.

Ongoing monitoring should include:

Quarterly retraining of the model to capture new hiring or attrition trends

Visualization dashboards to track cluster shifts over time

Feedback loops from HR managers to validate cluster interpretations

By embedding the model into HR systems, organizations can leverage data-driven segmentation to enhance decision-making in areas such as retention planning, workforce diversity, and employee engagement.

Double-click (or enter) to edit

```
# Import Various Libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import warnings

warnings.filterwarnings('ignore') # don't show me warnings going forward


# Load Dataset

hr_df = pd.read_csv('Human Resources.csv')

print('First 5 Heads Dataset')
print(hr_df.head())
print("-----------------------------------------------------------------")

```

```
# # Convert to Dataframe
print('Convert to DataFrame')
hr_df = pd.DataFrame(hr_df)
print(hr_df)
print("----------------------------------------------------------------------")


```

```
First 5 Heads Dataset
           id first_name    last_name  birthdate  gender  \
0  00-0037846      Kimmy   Walczynski   6/4/1991    Male
1  00-0041533   Ignatius    Springett  6/29/1984    Male
2  00-0045747     Corbie  Bittlestone  7/29/1989    Male
3  00-0055274       Baxy       Matton  9/14/1982  Female
4  00-0076100    Terrell         Suff  4/11/1994  Female

                       race            department  \
0         Hispanic or Latino           Engineering
1                      White  Business Development
2  Black or African American                 Sales
3                      White              Services
4          Two or More Races    Product Management

                     jobtitle      location   hire_date  \
0        Programmer Analyst I  Headquarters   1/20/2002
1            Business Analyst  Headquarters    4/8/2019
2  Solutions Engineer Manager  Headquarters  10/12/2010
3                Service Tech  Headquarters   4/10/2005
4            Business Analyst        Remote   9/29/2010

                  termdate location_city location_state
0                      NaN     Cleveland           Ohio
1                      NaN     Cleveland           Ohio
2                      NaN     Cleveland           Ohio
3                      NaN     Cleveland           Ohio
4  2029-10-29 06:09:38 UTC         Flint       Michigan
----------------------------------------------------------------------
Convert to DataFrame
               id first_name    last_name  birthdate  gender  \
0      00-0037846      Kimmy   Walczynski   6/4/1991    Male
1      00-0041533   Ignatius    Springett  6/29/1984    Male
2      00-0045747     Corbie  Bittlestone  7/29/1989    Male
3      00-0055274       Baxy       Matton  9/14/1982  Female
4      00-0076100    Terrell         Suff  4/11/1994  Female
...           ...        ...          ...        ...     ...
37403         NaN        NaN          NaN        NaN     NaN
37404         NaN        NaN          NaN        NaN     NaN
37405         NaN        NaN          NaN        NaN     NaN
37406         NaN        NaN          NaN        NaN     NaN
37407         NaN        NaN          NaN        NaN     NaN

                          race            department  \
0             Hispanic or Latino           Engineering
1                         White  Business Development
```

```
2      Black or African American                Sales
3                          White             Services
4              Two or More Races    Product Management
...                          ...                   ...
37403                      NaN                   NaN
37404                      NaN                   NaN
37405                      NaN                   NaN
37406                      NaN                   NaN
37407                      NaN                   NaN

                        jobtitle      location   hire_date  \
```

```
# # Inspect the Dataset:

print('Inspect the Dataset')
print(hr_df.info())
print("-------------------------------------------------------------------------")

print('Describe the Dataset')
print(hr_df.describe())
print("-------------------------------------------------------------------------")

print('Dataset Shape')
print(hr_df.shape)
print("-------------------------------------------------------------------------")

```

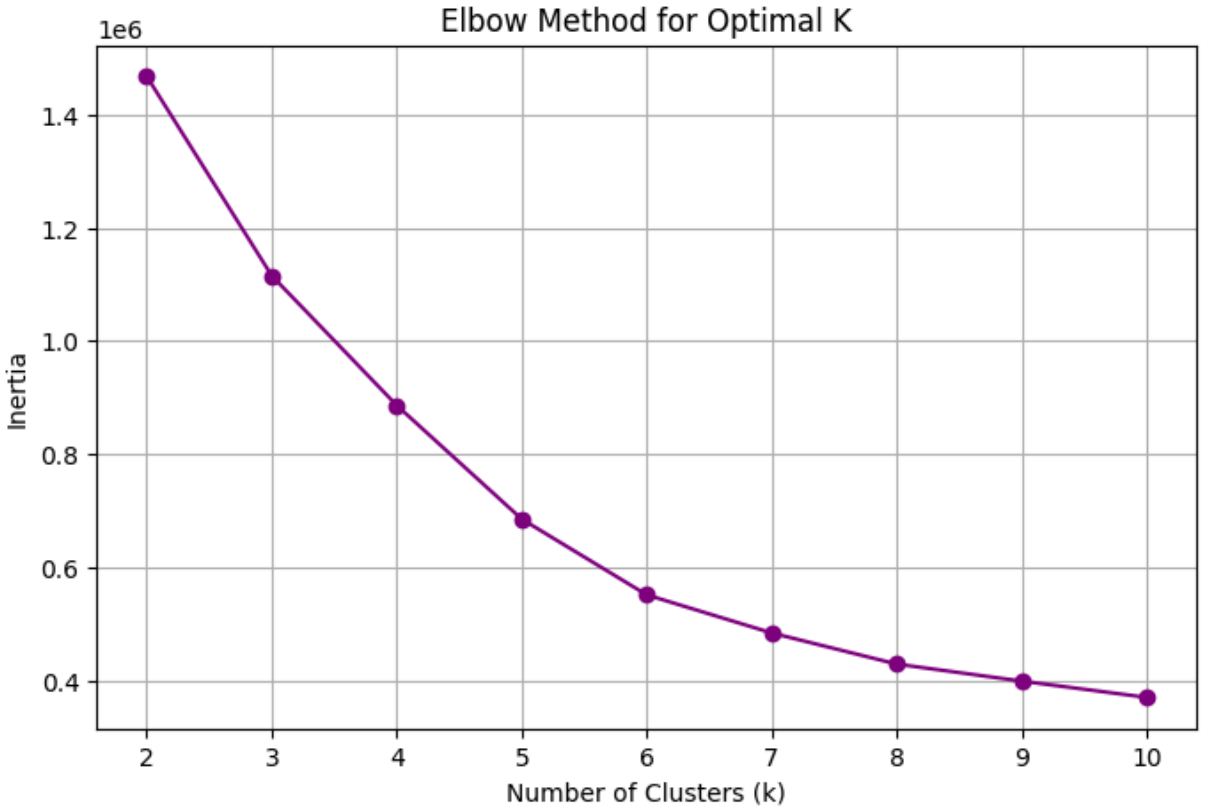
```
Inspect the Dataset
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37408 entries, 0 to 37407
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              22214 non-null  object
 1   first_name      22214 non-null  object
 2   last_name       22214 non-null  object
 3   birthdate       22214 non-null  object
 4   gender          22214 non-null  object
 5   race            22214 non-null  object
 6   department      22214 non-null  object
 7   jobtitle        22267 non-null  object
 8   location        22214 non-null  object
 9   hire_date       22214 non-null  object
 10  termdate        3929 non-null   object
 11  location_city   22214 non-null  object
 12  location_state  22214 non-null  object
dtypes: object(13)
memory usage: 3.7+ MB
None
-------------------------------------------------------------------------
Describe the Dataset
               id first_name last_name birthdate gender   race   department  \
count       22214      22214     22214     22214  22214  22214         22214
unique      22214       7758     17754     10854      3      7            13
```

```
top     99-9963543     Cassie    Ducker  5/9/1972   Male  White  Engineering
freq             1         12         7         9  11288   6328         6686

                     jobtitle      location hire_date  \
count                   22267         22214     22214
unique                    185             2      7016
top     Research Assistant II  Headquarters  3/4/2003
freq                      754         16715        10

                       termdate location_city location_state
count                      3929         22214          22214
unique                     3929            77              7
top     2012-12-10 14:29:59 UTC     Cleveland           Ohio
freq                          1         16871          18025
-----------------------------------------------------------------------
Dataset Shape
(37408, 13)
-----------------------------------------------------------------------
```

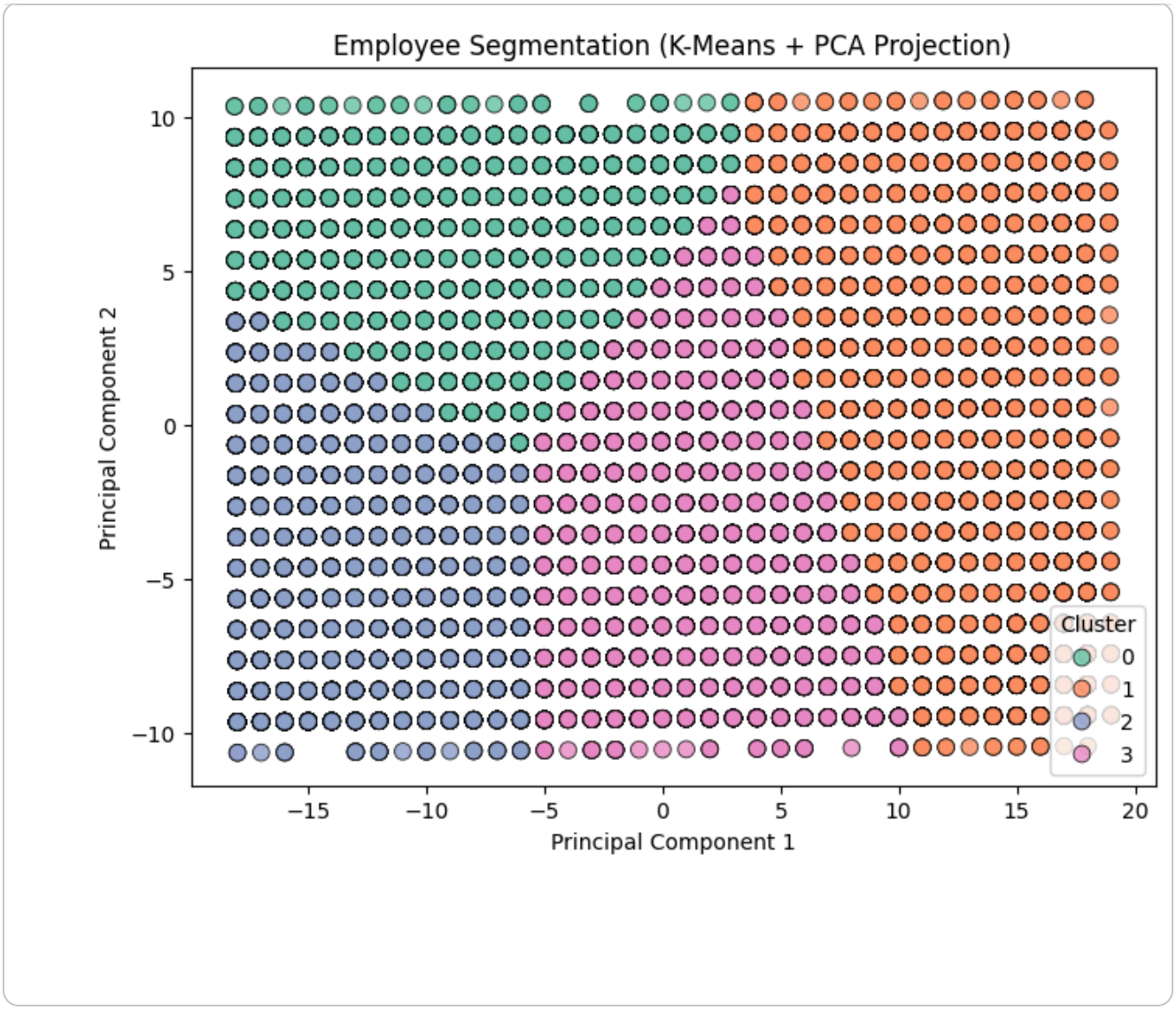
```
 1 from sklearn import cluster
 2 from sklearn.preprocessing import StandardScaler
 3
 4
 5 # # EDA
 6
 7 # print('Check Missing Values')
 8 # print(hr_df.isnull().sum())
 9 # print("-----------------------------------------------------------------
10
11 # View & Vizualize Missing Cols
12
13 # print('View & Vizualize Missing Cols')
14 # missing_perct_cols = (hr_df.isnull().sum()/hr_df.shape[0])*100
15 # print(missing_perct_cols)
16
17 # Viz - Using Heatmap to indicate the percentage of cols missing in my dat
18
19 # plt.figure(figsize=(10, 6))
20 # sns.heatmap(hr_df.isnull(), cbar=False, cmap='viridis')
21 # plt.title('Missing Values Heatmap')
22 # plt.show()
23
24 # Drop & Fill Missing Cols
25
26 # print('Drop Rows that are completely Empty')
27 # hr_df.dropna(how='all', inplace=True)
28 # print(hr_df)
29 # print("----------------------------------------")
30
31 # hr_df.drop(['id', 'first_name','last_name'], axis=1, inplace=True, error
32
33 # # Fill Missing Cols - Re-evaluating this section to ensure date columns
```

```
34
35 # print('Fill Missing Cols')
36 # hr_df['department'] = hr_df['department'].fillna('Unknown')
37 # print(hr_df)
38 # print("-----------------------------------------------------")
39
40 # print(hr_df.columns.tolist) #to check cols
41 # # print(hr_df['termdate'].unique) # to verify that the fill missing cols
42
43 # Convert Date Cols to Datetime
44
45 # hr_df['birthdate'] = pd.to_datetime(hr_df['birthdate'], errors='coerce')
46 # hr_df['hire_date'] = pd.to_datetime(hr_df['hire_date'], errors='coerce')
47 # hr_df['termdate'] = pd.to_datetime(hr_df['termdate'], errors='coerce') #
48
49 # Handle Missing Values in Date Columns AFTER Conversion
50
51 # For numerical calculations like age and years of service, fill with medi
52
53 # median_birth = hr_df['birthdate'].median() # Replace missing birthdate &
54 # median_hire = hr_df['hire_date'].median()
55
56 # hr_df['birthdate'].fillna(median_birth, inplace=True)
57 # hr_df['hire_date'].fillna(median_hire, inplace=True)
58
59 # # Saniity Check
60 # print("Number of missing birthdates after fill:", hr_df['birthdate'].isr
61 # print("Number of missing hire_dates after fill:", hr_df['hire_date'].isr
62
63
64 ##For termdate, where 'Active' was used, NaT represents 'Active' after coe
65
66 # Feature Engineering (to creates additional cols from the dataset for sig
67
68 # today = pd.to_datetime('today').replace(tzinfo=None)
69 # hr_df['age'] = (today - hr_df['birthdate']).dt.days // 365
70 # # Assuming 'hire_date' will now be in datetime format after dropping NaT
71 # hr_df['years_of_service'] = (today - hr_df['hire_date']).dt.days // 365
72
73 ## Adding Safety Check to ensure no NaT remain in my numerical syntax:
74
75 # hr_df['age'] = hr_df['age'].fillna(hr_df['age'].median())
76 # hr_df['years_of_service'] = hr_df['years_of_service'].fillna(hr_df['year
77
78 # Create new employment status column based on termdate NaT values
79
80 # hr_df['employment_status'] = hr_df['termdate'].apply(lambda x: 'Active'
81
82
83 # Drop any date_time cols:
84
```

```
# # Drop datetime columns that were used to calculate numeric features
# hr_df.drop(columns=['birthdate', 'hire_date', 'termdate'], inplace=True,

# Verify no datetime columns remain in my dataset:

# print("Remaining datetime columns:", hr_df.select_dtypes(include=['datet

# Ecnode and Scale the Dataset:

# -- Select the Categorical Dataset ---

categorical_cols = ['gender','race','department','jobtitle','location_city
numerical_cols = ['age','years_of_service']

# # -- One-Hot Encode Categorical Cols ---

# Applying one-hot encoding to the specified categorical columns
hot_codes = pd.get_dummies(hr_df, columns=categorical_cols, drop_first=Tru


# # --- Scale Numerical Features---using Scaler function from Sklearn
# scaler = StandardScaler()
# hot_codes[numerical_cols] = scaler.fit_transform(hot_codes[numerical_col

# print("Conversion of Hot Encoding & Scale Codes:", hot_codes.shape)
# print(hot_codes.head())

# # Conduct Clustering on my dataset:

inertia = []
Cluster = range(2,11) # Use range to iterate from 2 to 10 clusters

for cluster in Cluster:
    k_means = KMeans(n_clusters=cluster, random_state=42, n_init=10) # Adc
    k_means.fit(hot_codes)
    inertia.append(k_means.inertia_)


# Plot Elbow Curve
plt.figure(figsize=(8,5))
plt.plot(Cluster, inertia, 'o-', color='purple')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.grid(True)
plt.show()
```




```
from sklearn.decomposition import PCA

# --- PCA Transformation ---
pca = PCA(n_components=2)
pca_result = pca.fit_transform(hot_codes)

# Create a DataFrame for PCA components and clusters
pca_df = pd.DataFrame(pca_result, columns=['PCA1', 'PCA2'])
pca_df['Cluster'] = hr_df['KMeans_Cluster']

# --- Improved PCA Visualization ---
plt.figure(figsize=(8,6))
sns.scatterplot(
    x='PCA1', y='PCA2',
    hue='Cluster', data=pca_df,
    palette='Set2', s=60, alpha=0.8, edgecolor='k'
)
plt.title("Employee Segmentation (K-Means + PCA Projection)")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.legend(title='Cluster', loc='best')
plt.show()

```



Results Interpretation for the above Cluserings / Graphs (K-MEANs & PCA)

The Elbow Method graph showed a clear inflection point at k = 4, suggesting that four clusters provide the most meaningful segmentation of the HR dataset. Beyond four clusters, the reduction in inertia became marginal, confirming that additional clusters would not significantly improve model performance. Thus, k = 4 was selected as the optimal number of employee groups for analysis.

After applying K-Means clustering with k = 4 and visualizing through PCA, distinct employee clusters emerged. The PCA projection revealed well-separated regions, indicating that the selected features—such as age, years of service, department, job title, and employment status—contribute effectively to distinguishing employee profiles. The clusters show clear differentiation, with each group representing unique employee characteristics and work patterns.

In practical terms, these clusters may reflect different workforce segments—for example, long-tenured employees nearing retirement, mid-career staff with stable performance, new hires or early-career employees, and remote or specialized roles. HR managers could leverage these insights to tailor engagement and retention strategies to each group, improve onboarding for new hires, and design development programs that address the needs of mid-career employees. Overall, the clustering results provide a data-driven foundation for more targeted and equitable workforce planning.

```
from sklearn.metrics import silhouette_score

# Compute Silhouette Score for the final model
silhouette_avg = silhouette_score(hot_codes, hr_df['KMeans_Cluster'])
print(f"Silhouette Score: {silhouette_avg:.3f}")


# Evaluate multiple cluster counts for comparison

for k in range(2, 7):
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(hot_codes)
    score = silhouette_score(hot_codes, labels)
    print(f"k={k}: Silhouette Score = {score:.3f}")

```

```
Silhouette Score: 0.322
k=2: Silhouette Score = 0.456
k=3: Silhouette Score = 0.373
k=4: Silhouette Score = 0.322
k=5: Silhouette Score = 0.333
k=6: Silhouette Score = 0.341
```

At first, when I ran the silhouette score and got 0.3222. I needed to compare the score using other clusters based on a range, which I was aiming for a socre closer to +1. However, the additional Silhouette Scores indicate how well the data points fit within their assigned clusters for different numbers of clusters (k)

The scores are reflective of how distinctly employees can be grouped based on their demographic and job-related features. In real-life terms, the higher score at k=2 (0.456) may represent two broad employee categories — for example, new hires versus long-tenured staff.

As the number of clusters increases (k=3–6), scores slightly decline, meaning the model starts identifying smaller subgroups such as mid-career employees, remote workers, or specialized job roles, which naturally overlap in characteristics. Choosing k=4 (0.322) captures this complexity realistically — mirroring how organizations often have multiple

overlapping workforce segments rather than completely distinct groups. These clusters can be used by HR teams to design targeted initiatives such as onboarding programs for new hires, career development plans for mid-level employees, or retention strategies for experienced staff.