This is the upyter Notebook Used for working on Data Science Assignment by Hamza Rehman 028

In [1]:
```
import pandas as pd
```

Question 1

In [89]:
```
def label_row(row):
    # Convert the column to numeric, coercing errors to NaN
    impact_factor = pd.to_numeric(row['Journal Impact Factor'], errors='coer

    if pd.isna(impact_factor):
        return "Unknown"  # Handle cases where the conversion failed

    if impact_factor > 5:
        return "Outstanding"
    elif impact_factor > 3:
        return "Good"
    elif impact_factor > 1:
        return "Fair"
    else:
        return "Very Poor"

def label_instances(data):
    # Apply the labeling logic to each row
    data['Label'] = data.apply(label_row, axis=1)
    return data

# Example of how to load the data and apply the function
def main_part1():
    # Assuming 'filepath' is the path to your Excel dataset
    filepath = '2017 Impact Factor Journals - Release in 2018.xlsx'

    # Use pd.read_excel to load the Excel file
    data = pd.read_excel(filepath)

    # Label the dataset
    labeled_data = label_instances(data)

    if 'Label' in labeled_data.columns:
        print("\nLabeled Data:")
        print(labeled_data.head())
    else:
        print("Labeling could not be completed. Please check the dataset.")

if __name__ == "__main__":
    main_part1()
```

```
Labeled Data:
  Rank                 Full Journal Title Total Cites  Unnamed: 3  \
0    1  CA-A CANCER JOURNAL FOR CLINICIANS      28,839         NaN
1    2     NEW ENGLAND JOURNAL OF MEDICINE     332,830         NaN
2    3                              LANCET     233,269         NaN
3    4                   CHEMICAL REVIEWS     174,920         NaN
4    5            Nature Reviews Materials       3,218         NaN

  Journal Impact Factor  Eigenfactor Score        Label
0               244.585            0.06603  Outstanding
1                79.258            0.70200  Outstanding
2                53.254            0.43574  Outstanding
3                52.613            0.26565  Outstanding
4                51.941            0.01506  Outstanding
```

Data Before Labels

In [21]:
```
data
```

Out[21]:

	Rank	Full Journal Title	Total Cites	Unnamed: 3	Journal Impact Factor	Eigenfactor Score
0	1	CA-A CANCER JOURNAL FOR CLINICIANS	28,839	NaN	244.585	0.06603
1	2	NEW ENGLAND JOURNAL OF MEDICINE	332,830	NaN	79.258	0.70200
2	3	LANCET	233,269	NaN	53.254	0.43574
3	4	CHEMICAL REVIEWS	174,920	NaN	52.613	0.26565
4	5	Nature Reviews Materials	3,218	NaN	51.941	0.01506
...	...	...	...	...	...	...
12297	12271	SLAS Discovery	96	NaN	Not Available	0.00000
12298	12271	SLAS Technology	49	NaN	Not Available	0.00000
12299	12271	Sustainable Energy & Fuels	166	NaN	Not Available	0.00000
12300	Copyright © 2018 Clarivate Analytics	NaN	NaN	NaN	NaN	NaN
12301	By exporting the selected data, you agree to t...	NaN	NaN	NaN	NaN	NaN

12302 rows × 6 columns

Data After Labels

In [33]:
```
labeled_data = labeled_data.drop(columns=['Unnamed: 3'], errors='ignore')  #
labeled_data['Journal Impact Factor'] = labeled_data['Journal Impact Factor'
```

In [36]:
```
labeled_data
```

Out[36]:

	Rank	Full Journal Title	Total Cites	Journal Impact Factor	Eigenfactor Score	Label
0	1	CA-A CANCER JOURNAL FOR CLINICIANS	28,839	244.585	0.06603	Outstanding
1	2	NEW ENGLAND JOURNAL OF MEDICINE	332,830	79.258	0.70200	Outstanding
2	3	LANCET	233,269	53.254	0.43574	Outstanding
3	4	CHEMICAL REVIEWS	174,920	52.613	0.26565	Outstanding
4	5	Nature Reviews Materials	3,218	51.941	0.01506	Outstanding
...	...	...	...	...	...	...
12297	12271	SLAS Discovery	96	Not Available	0.00000	Unknown
12298	12271	SLAS Technology	49	Not Available	0.00000	Unknown
12299	12271	Sustainable Energy & Fuels	166	Not Available	0.00000	Unknown
12300	Copyright © 2018 Clarivate Analytics	NaN	NaN	NaN	NaN	Unknown
12301	By exporting the selected data, you agree to t...	NaN	NaN	NaN	NaN	Unknown

12302 rows × 6 columns

Question 2

In [39]:

```
import pandas as pd

# Part 2: Classify New Instances
def classify_new_instances(new_instances, criteria):
    labeled_instances = []
    for instance in new_instances:
        if instance['Journal Impact Factor'] > criteria['Outstanding']:
```

```
            labeled_instances.append("Outstanding")
        elif instance['Journal Impact Factor'] > criteria['Good']:
            labeled_instances.append("Good")
        elif instance['Journal Impact Factor'] > criteria['Weak']:
            labeled_instances.append("Weak")
        else:
            labeled_instances.append("Very Poor")
    return labeled_instances

def main_part2():
    new_instances = pd.DataFrame([
        {'Journal Impact Factor': float(input("Enter Impact Factor for insta
        {'Journal Impact Factor': float(input("Enter Impact Factor for insta
        {'Journal Impact Factor': float(input("Enter Impact Factor for insta
        {'Journal Impact Factor': float(input("Enter Impact Factor for insta
        {'Journal Impact Factor': float(input("Enter Impact Factor for insta
        {'Journal Impact Factor': float(input("Enter Impact Factor for insta
        {'Journal Impact Factor': float(input("Enter Impact Factor for insta
        {'Journal Impact Factor': float(input("Enter Impact Factor for insta
        {'Journal Impact Factor': float(input("Enter Impact Factor for insta
        {'Journal Impact Factor': float(input("Enter Impact Factor for insta
    ])

    criteria = {'Outstanding': 5, 'Good': 3, 'Weak': 1}  # Define thresholds
    new_labels = classify_new_instances(new_instances.to_dict('records'), cr
    print("\nNew Instance Labels:")
    print(new_labels)

if __name__ == "__main__":
    main_part2()
```

```
New Instance Labels:
['Weak', 'Good', 'Very Poor', 'Good', 'Good', 'Weak', 'Very Poor', 'Weak',
'Weak', 'Good']
```

Question 3

In [63]:
```
filepath = '2017 Impact Factor Journals - Release in 2018.xlsx'
data = pd.read_excel(filepath)
```

In [64]:
```
data.head()
```

Out[64]:

	Rank	Full Journal Title	Total Cites	Unnamed: 3	Journal Impact Factor	Eigenfactor Score
0	1	CA-A CANCER JOURNAL FOR CLINICIANS	28,839	NaN	244.585	0.06603
1	2	NEW ENGLAND JOURNAL OF MEDICINE	332,830	NaN	79.258	0.70200
2	3	LANCET	233,269	NaN	53.254	0.43574
3	4	CHEMICAL REVIEWS	174,920	NaN	52.613	0.26565
4	5	Nature Reviews Materials	3,218	NaN	51.941	0.01506

In [65]:
```
data = data.drop(columns=['Unnamed: 3'], errors='ignore')
data['Journal Impact Factor'] = data['Journal Impact Factor'].replace({',':
```

In [66]:
```
data.head()
```

Out[66]:

	Rank	Full Journal Title	Total Cites	Journal Impact Factor	Eigenfactor Score
0	1	CA-A CANCER JOURNAL FOR CLINICIANS	28,839	244.585	0.06603
1	2	NEW ENGLAND JOURNAL OF MEDICINE	332,830	79.258	0.70200
2	3	LANCET	233,269	53.254	0.43574
3	4	CHEMICAL REVIEWS	174,920	52.613	0.26565
4	5	Nature Reviews Materials	3,218	51.941	0.01506

In [53]:
```
  data_cleaned = data.dropna()
features = data_cleaned[['Journal Impact Factor']]
```

In [72]:
```
data_cleaned['Journal Impact Factor'] = pd.to_numeric(data_cleaned['Journal

print(data_cleaned.dtypes)

print(data_cleaned['Journal Impact Factor'].isnull().sum())
```

```
Rank                    object
Full Journal Title      object
Total Cites             object
Journal Impact Factor   float64
Eigenfactor Score       float64
dtype: object
30
```

Previous Work was done for my own Understanding

Code for question 3 Starts now

In [87]:
```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

def perform_clustering(data, n_clusters=3):
    features = data[['Journal Impact Factor', 'Total Cites', 'Eigenfactor Sc

    scaler = StandardScaler()
    scaled_features = scaler.fit_transform(features)

    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    data['Cluster'] = kmeans.fit_predict(scaled_features)

    return data, kmeans, scaler

def main_part3():
    filepath = '2017 Impact Factor Journals - Release in 2018.xlsx'
    data = pd.read_excel(filepath)

    # Drop the 'Unnamed: 3' column
    if 'Unnamed: 3' in data.columns:
        data.drop(columns=['Unnamed: 3'], inplace=True)

    # Inspect the number of missing values in each column
    print("Missing Values Before Cleaning:")
    print(data.isnull().sum())

    # Convert columns to numeric and fill missing values with the column's m
    data['Journal Impact Factor'] = pd.to_numeric(data['Journal Impact Factc
    data['Total Cites'] = pd.to_numeric(data['Total Cites'], errors='coerce'
    data['Eigenfactor Score'] = pd.to_numeric(data['Eigenfactor Score'], err

    # Fill NaN values with the column's mean
    data['Journal Impact Factor'].fillna(data['Journal Impact Factor'].mean(
    data['Total Cites'].fillna(data['Total Cites'].mean(), inplace=True)
    data['Eigenfactor Score'].fillna(data['Eigenfactor Score'].mean(), inpla

    # Check if data is cleaned properly now
    print("\nMissing Values After Cleaning:")
    print(data.isnull().sum())

    # Remove rows with NaN values in the relevant columns only
    data_cleaned = data.dropna(subset=['Journal Impact Factor', 'Total Cites

    # Print to check if data is still empty after cleaning
    print("\nData after cleaning:")
    print(data_cleaned[['Journal Impact Factor', 'Total Cites', 'Eigenfactor

    # Ensure that there are still rows to cluster
    if data_cleaned.empty:
```

```
        print("No data available for clustering after cleaning.")
        return

    # Perform clustering with the cleaned data
    clustered_data, _, _ = perform_clustering(data_cleaned)

    # Print the clustered data
    print("\nClustered Data (First Few Rows):")
    print(clustered_data[['Journal Impact Factor', 'Cluster']].head())

    # Print the cluster distribution
    print("\nCluster Distribution:")
    print(clustered_data['Cluster'].value_counts())

if __name__ == "__main__":
    main_part3()
```

```
Missing Values Before Cleaning:
Rank                      0
Full Journal Title        2
Total Cites               2
Journal Impact Factor     2
Eigenfactor Score         2
dtype: int64

Missing Values After Cleaning:
Rank                      0
Full Journal Title        2
Total Cites               0
Journal Impact Factor     0
Eigenfactor Score         0
dtype: int64

Data after cleaning:
   Journal Impact Factor  Total Cites  Eigenfactor Score
0                244.585   479.913477            0.06603
1                 79.258   479.913477            0.70200
2                 53.254   479.913477            0.43574
3                 52.613   479.913477            0.26565
4                 51.941   479.913477            0.01506

Clustered Data (First Few Rows):
   Journal Impact Factor  Cluster
0                244.585        2
1                 79.258        2
2                 53.254        2
3                 52.613        2
4                 51.941        2

Cluster Distribution:
Cluster
0    10560
1     1697
2       45
Name: count, dtype: int64
```

```
C:\Users\Hamza Rehman\AppData\Local\Temp\ipykernel_6796\3914417889.py:34: Fu
tureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behave
s as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'd
f.method({col: value}, inplace=True)' or df[col] = df[col].method(value) ins
tead, to perform the operation inplace on the original object.


  data['Journal Impact Factor'].fillna(data['Journal Impact Factor'].mean(),
inplace=True)
C:\Users\Hamza Rehman\AppData\Local\Temp\ipykernel_6796\3914417889.py:35: Fu
tureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behave
s as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'd
f.method({col: value}, inplace=True)' or df[col] = df[col].method(value) ins
tead, to perform the operation inplace on the original object.


  data['Total Cites'].fillna(data['Total Cites'].mean(), inplace=True)
C:\Users\Hamza Rehman\AppData\Local\Temp\ipykernel_6796\3914417889.py:36: Fu
tureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behave
s as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'd
f.method({col: value}, inplace=True)' or df[col] = df[col].method(value) ins
tead, to perform the operation inplace on the original object.


  data['Eigenfactor Score'].fillna(data['Eigenfactor Score'].mean(), inplace
=True)
```

Question 4

In [85]:

```python
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

data = data_cleaned

# Select the relevant features for clustering
features = data[['Journal Impact Factor', 'Total Cites', 'Eigenfactor Score'

# Scale the features using StandardScaler
scaler = StandardScaler()
```

```
scaled_features = scaler.fit_transform(features)

n_clusters = 3
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
data['Predicted Cluster'] = kmeans.fit_predict(scaled_features)


new_instances = pd.DataFrame({
    'Journal Impact Factor': [65.2, 59.1, 95.3, 42.7, 68.8, 52.9, 88.1, 74.5
    'Total Cites': [460.0, 510.0, 475.0, 530.0, 490.0, 455.0, 495.0, 475.0,
    'Eigenfactor Score': [0.55, 0.62, 0.45, 0.72, 0.52, 0.59, 0.72, 0.54, 0.
})

scaled_new_instances = scaler.transform(new_instances)

predicted_clusters = kmeans.predict(scaled_new_instances)

new_instances['Predicted Cluster'] = predicted_clusters

print(new_instances)
```

	Journal Impact Factor	Total Cites	Eigenfactor Score	Predicted Cluster
0	65.2	460.0	0.55	0
1	59.1	510.0	0.62	0
2	95.3	475.0	0.45	1
3	42.7	530.0	0.72	0
4	68.8	490.0	0.52	0
5	52.9	455.0	0.59	0
6	88.1	495.0	0.72	0
7	74.5	475.0	0.54	0
8	62.3	465.0	0.43	1
9	78.0	480.0	0.61	0

```
C:\Users\Hamza Rehman\Desktop\Sample\env\Lib\site-packages\sklearn\cluster\_
kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Window
s with MKL, when there are less chunks than available threads. You can avoid
it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```