```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES
# TO THE CORRECT LOCATION (/kaggle/input) IN YOUR NOTEBOOK,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.

import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil

CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = 'facebook-live-sellers-in-thailand-uci-ml-repo:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2

KAGGLE_INPUT_PATH='/kaggle/input'
KAGGLE_WORKING_PATH='/kaggle/working'
KAGGLE_SYMLINK='kaggle'

!umount /kaggle/input/ 2> /dev/null
shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)

try:
  os.symlink(KAGGLE_INPUT_PATH, os.path.join("..", 'input'), target_is_directory=True)
except FileExistsError:
  pass
try:
  os.symlink(KAGGLE_WORKING_PATH, os.path.join("..", 'working'), target_is_directory=True)
except FileExistsError:
  pass

for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
    directory, download_url_encoded = data_source_mapping.split(':')
    download_url = unquote(download_url_encoded)
    filename = urlparse(download_url).path
    destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
    try:
        with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
            total_length = fileres.headers['content-length']
            print(f'Downloading {directory}, {total_length} bytes compressed')
            dl = 0
            data = fileres.read(CHUNK_SIZE)
            while len(data) > 0:
                dl += len(data)
                tfile.write(data)
                done = int(50 * dl / int(total_length))
                sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl} bytes downloaded")
                sys.stdout.flush()
                data = fileres.read(CHUNK_SIZE)
            if filename.endswith('.zip'):
              with ZipFile(tfile) as zfile:
                zfile.extractall(destination_path)
            else:
              with tarfile.open(tfile.name) as tarfile:
                tarfile.extractall(destination_path)
            print(f'\nDownloaded and uncompressed: {directory}')
    except HTTPError as e:
        print(f'Failed to load (likely expired) {download_url} to path {destination_path}')
        continue
    except OSError as e:
        print(f'Failed to load {download_url} to path {destination_path}')
        continue

print('Data source import complete.')
```

```
Downloading facebook-live-sellers-in-thailand-uci-ml-repo, 141194 bytes compressed
[==================================================] 141194 bytes downloaded
Downloaded and uncompressed: facebook-live-sellers-in-thailand-uci-ml-repo
Data source import complete.
```

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
```

```
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a versi
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/facebook-live-sellers-in-thailand-uci-ml-repo/Live.csv
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


book_data=pd.read_csv('Live.csv')
book_df=pd.DataFrame(book_data)
book_df.head()
```

	status_id	status_type	status_published	num_reactions
0	246675545449582_1649696485147474	video	4/22/2018 6:00	529
1	246675545449582_1649426988507757	photo	4/21/2018 22:45	150
2	246675545449582_1648730588577397	video	4/21/2018 6:17	227
3	246675545449582_1648576705259452	photo	4/21/2018 2:29	111
4	246675545449582_1645700502213739	photo	4/18/2018 3:22	213

Next steps: Generate code with book_df View recommended plots

```
book_df.info
```

```
pandas.core.frame.DataFrame.info
def info(verbose: bool | None=None, buf: WriteBuffer[str] | None=None,
max_cols: int | None=None, memory_usage: bool | str | None=None,
show_counts: bool | None=None) -> None
```

/usr/local/lib/python3.10/dist-packages/pandas/core/frame.py
Print a concise summary of a DataFrame.

This method prints information about a DataFrame including
the index dtype and columns, non-null values and memory usage.

```
book_df.describe()
```

	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wo
count	7050.000000	7050.000000	7050.000000	7050.000000	7050.000000	7050.0000
mean	230.117163	224.356028	40.022553	215.043121	12.728652	1.2893
std	462.625309	889.636820	131.599965	449.472357	39.972930	8.7196
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	17.000000	0.000000	0.000000	17.000000	0.000000	0.0000
50%	59.500000	4.000000	0.000000	58.000000	0.000000	0.0000
75%	219.000000	23.000000	4.000000	184.750000	3.000000	0.0000
max	4710.000000	20990.000000	3424.000000	4710.000000	657.000000	278.0000

Data cleaning checking for missing values, duplicate values , dropping irrelevant columns

```
book_df.isnull().sum()
```

```
status_id              0
status_type            0
status_published       0
num_reactions          0
num_comments           0
num_shares             0
num_likes              0
num_loves              0
num_wows               0
num_hahas              0
num_sads               0
num_angrys             0
Column1             7050
Column2             7050
Column3             7050
Column4             7050
dtype: int64
```

```
book_df.duplicated().sum()
```

```
51
```

```
book_df.duplicated()
```

```
0       False
1       False
2       False
3       False
4       False
        ...
7045    False
7046    False
7047    False
7048    False
7049    False
Length: 7050, dtype: bool
```

```
book_df_new=book_df.drop_duplicates()
book_df_new.info
```

```
pandas.core.frame.DataFrame.info
def info(verbose: bool | None=None, buf: WriteBuffer[str] | None=None,
max_cols: int | None=None, memory_usage: bool | str | None=None,
show_counts: bool | None=None) -> None
```

/usr/local/lib/python3.10/dist-packages/pandas/core/frame.py

```
Print a concise summary of a DataFrame.

This method prints information about a DataFrame including
the index dtype and columns, non-null values and memory usage.
```

```
book_df_new1=book_df_new.drop(["Column1",'Column2','Column3','Column4',"status_id"],axis=1)
```

```
book_df_new1.head()
```

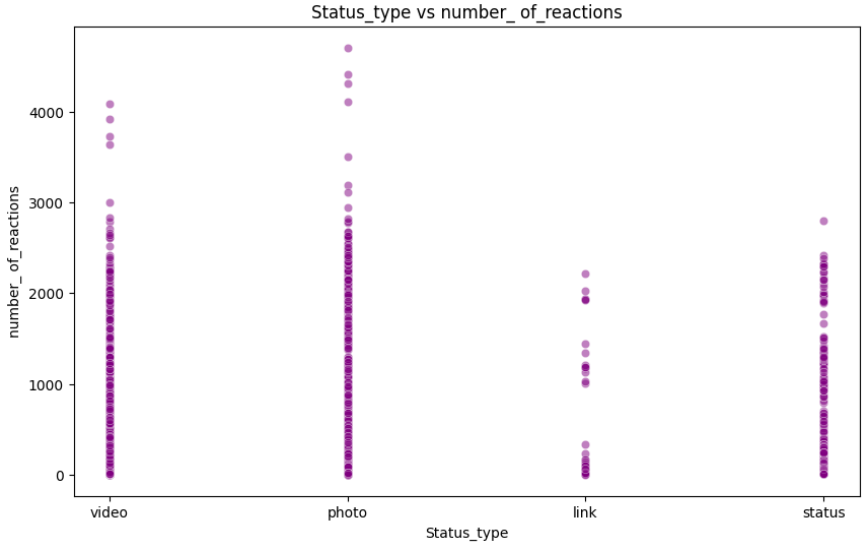
	status_type	status_published	num_reactions	num_comments	num_shares	num_
0	video	4/22/2018 6:00	529	512	262	
1	photo	4/21/2018 22:45	150	0	0	
2	video	4/21/2018 6:17	227	236	57	
3	photo	4/21/2018 2:29	111	0	0	
4	photo	4/18/2018 3:22	213	0	0	

Next steps: Generate code with book_df_new1 View recommended plots

Our data is clean now and ready for visualization with matplotlib and seaborn libraries

```
plt.figure(figsize=(10,6))
sns.scatterplot(x=book_df_new1["status_type"],y=book_df_new1["num_reactions"],color="purple",alpha=0.5)
plt.title("Status_type vs number_ of_reactions")
plt.xlabel("Status_type")
plt.ylabel("number_ of_reactions")
```

```
Text(0, 0.5, 'number_ of_reactions')
```




```
ordinal_map={'video':1,
             'photo':2,
              'link':3,
               'status':4


}
book_df_new1["status_encoded"]=book_df_new1.status_type.map(ordinal_map)
book_df_new1.head()
```

	status_type	status_published	num_reactions	num_comments	num_shares	num_
0	video	4/22/2018 6:00	529	512	262	
1	photo	4/21/2018 22:45	150	0	0	
2	video	4/21/2018 6:17	227	236	57	
3	photo	4/21/2018 2:29	111	0	0	
4	photo	4/18/2018 3:22	213	0	0	

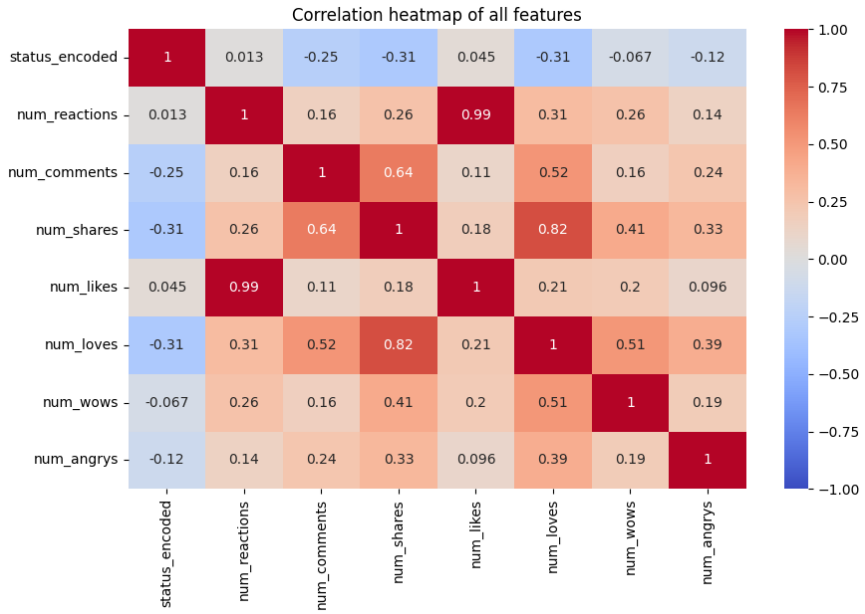
Next steps: Generate code with book_df_new1 View recommended plots

```
Start coding or generate with AI.
```

```
book_df_new1.columns
```

```
Index(['status_type', 'status_published', 'num_reactions', 'num_comments',
       'num_shares', 'num_likes', 'num_loves', 'num_wows', 'num_hahas',
       'num_sads', 'num_angrys', 'status_encoded'],
      dtype='object')
```

```
features= ['status_encoded',"num_reactions","num_comments","num_shares",'num_likes',"num_loves","num_wows","num_angrys"]
corr_matrix= book_df_new1 [features].corr()
plt.figure(figsize=(10,6))
sns.heatmap(corr_matrix,annot=True,cmap="coolwarm",vmin=-1,vmax=1)
plt.title("Correlation heatmap of all features")
plt.show()
```




```
print(corr_matrix['status_encoded'].sort_values(ascending=False))
```

```
status_encoded    1.000000
num_likes         0.045070
num_reactions     0.013094
num_wows         -0.067470
num_angrys       -0.118767
num_comments     -0.254606
num_loves        -0.313025
num_shares       -0.314830
Name: status_encoded, dtype: float64
```

```
book_df_new2=book_df_new1.drop(["status_type","status_published"],axis=1)
book_df_new2.head()
```

	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num
0	529	512	262	432	92	3	
1	150	0	0	150	0	0	
2	227	236	57	204	21	1	
3	111	0	0	111	0	0	
4	213	0	0	204	9	0	

Next steps: Generate code with book_df_new2 View recommended plots

```
# application of k-means clustering algorithim

from sklearn.preprocessing import MinMaxScaler

ms = MinMaxScaler()

X = ms.fit_transform(book_df_new2)

from sklearn.cluster import KMeans

import warnings

warnings.filterwarnings('ignore')

from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(book_df_new2)

# Apply K-means clustering
k = 3  # Number of clusters
kmeans = KMeans(n_clusters=k, init='k-means++', n_init=10, random_state=42)
kmeans.fit(X_scaled)

# Get cluster labels
labels = kmeans.labels_


kmeans=KMeans(n_clusters=3,random_state=0)
kmeans.fit(X_scaled)
labels=kmeans.labels_
book_df_new2["Cluster"]=labels
book_df_new2.head()
```

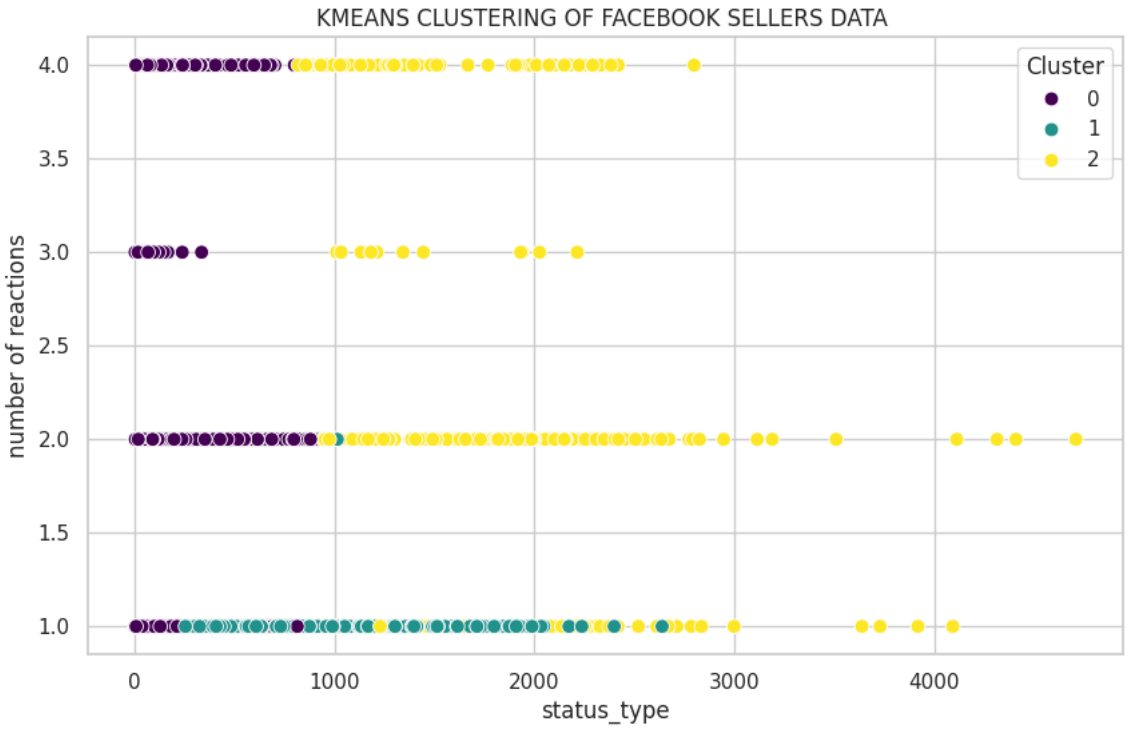
	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num
0	529	512	262	432	92	3	
1	150	0	0	150	0	0	
2	227	236	57	204	21	1	
3	111	0	0	111	0	0	
4	213	0	0	204	9	0	

Next steps: Generate code with book_df_new2 View recommended plots

```
data=book_df_new2


sns.set(style="whitegrid")
plt.figure(figsize=(10,6))
sns.scatterplot(x=book_df_new2["status_encoded"],y=book_df_new2["num_reactions"],hue=book_df_new2["Cluster"],palette="viridis
plt.title("KMEANS CLUSTERING OF FACEBOOK SELLERS DATA ")
plt.xlabel("status_type")
plt.ylabel("number of reactions")
plt.legend(title="Cluster")
plt.show()
```


```
sns.set(style="whitegrid")
plt.figure(figsize=(10,6))
sns.scatterplot(x=book_df_new2["num_reactions"],y=book_df_new2["status_encoded"],hue=book_df_new2["Cluster"],palette="viridis"
plt.title("KMEANS CLUSTERING OF FACEBOOK SELLERS DATA ")
plt.xlabel("status_type")
plt.ylabel("number of reactions")
plt.legend(title="Cluster")
```

```
plt.show()
```



Double-click (or enter) to edit

```
kmeans.cluster_centers_
```

```
array([[-0.2568456 , -0.14485562, -0.15105935, -0.24877262, -0.14195825,
        -0.08445882, -0.09520731, -0.06979614, -0.08822324,  0.00742703],
       [ 1.11622975,  3.17803299,  3.34269316,  0.79889782,  3.19990747,
         1.59816585,  2.10548713,  1.59197772,  1.9810748 , -1.03391009],
       [ 3.40184513, -0.18819161, -0.21985634,  3.52824538, -0.25505854,
         0.10091088, -0.13749884, -0.14085399, -0.1522444 ,  0.72964763]])
```

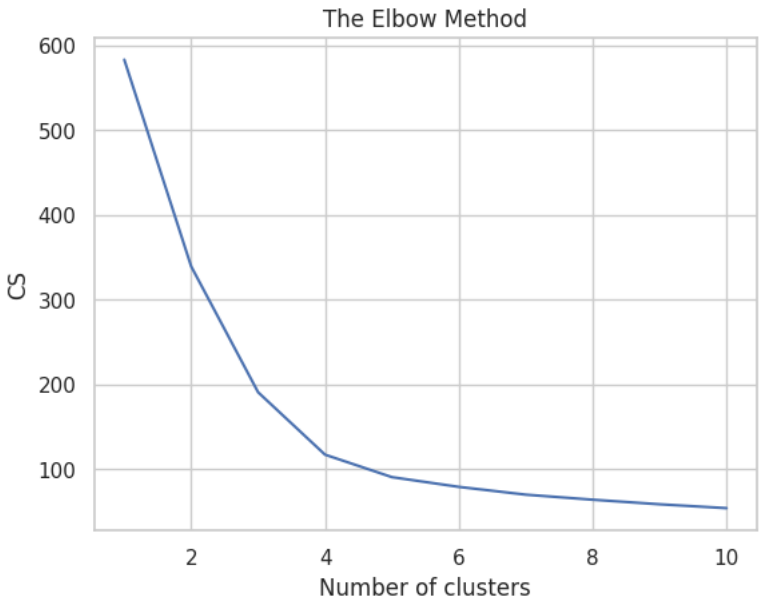
```
kmeans.inertia_
```

```
44453.16590929941
```

```
kmeans = KMeans(n_clusters=3,init='k-means++', random_state=0)

kmeans.fit(book_df_new2)

# check how many of the samples were correctly labeled
labels = kmeans.labels_

correct_labels = sum(book_df_new2["status_encoded"] == labels)
print("Result: %d out of %d samples were correctly labeled." % (correct_labels, book_df_new2["status_encoded"].size))
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(book_df_new2["status_encoded"].size)))
```

```
Result: 2030 out of 6999 samples were correctly labeled.
Accuracy score: 0.29
```

```
from sklearn.cluster import KMeans
cs = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit(X)
    cs.append(kmeans.inertia_)
plt.plot(range(1, 11), cs)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('CS')
plt.show()
```




```
kmeans=KMeans(n_clusters=2,random_state=0)
kmeans.fit(book_df_new2)
labels=kmeans.labels_
book_df_new2["Cluster"]=labels
book_df_new2.head()
```

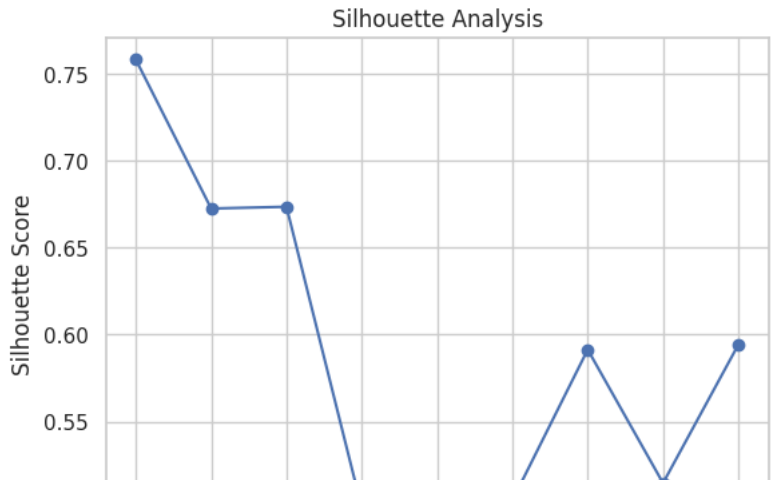
	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_hahas	num_sads	num_angrys	status_encoded
0	529	512	262	432	92	3	1	1	0	1
1	150	0	0	150	0	0	0	0	0	2
2	227	236	57	204	21	1	1	0	0	1
3	111	0	0	111	0	0	0	0	0	2
4	213	0	0	204	9	0	0	0	0	2

Next steps: Generate code with book_df_new2 View recommended plots

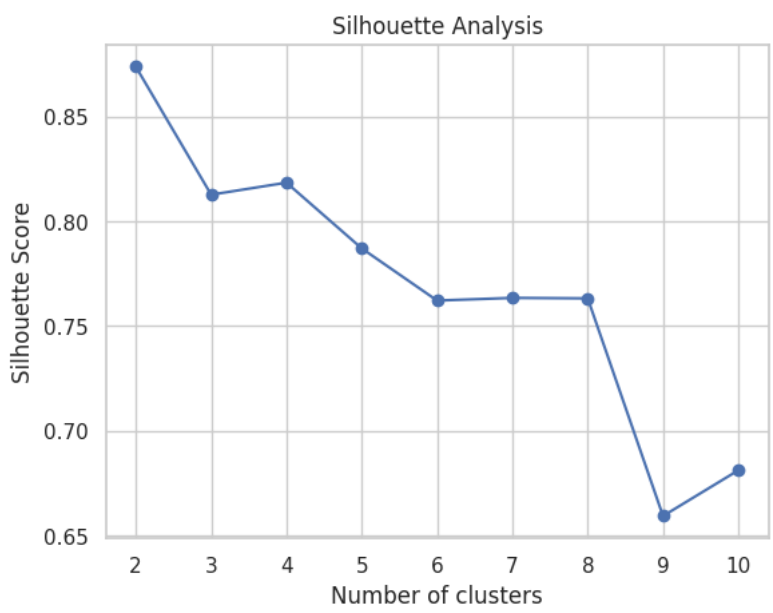
```
kmeans = KMeans(n_clusters=2,init='k-means++', random_state=0)

kmeans.fit(book_df_new2)

# check how many of the samples were correctly labeled
labels = kmeans.labels_

correct_labels = sum(book_df_new2["status_encoded"] == labels)
print("Result: %d out of %d samples were correctly labeled." % (correct_labels, book_df_new2["status_encoded"].size))
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(book_df_new2["status_encoded"].size)))
```

```
Result: 140 out of 6999 samples were correctly labeled.
Accuracy score: 0.02
```

```
from sklearn.metrics import silhouette_score

silhouette_scores = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, init='k-means++', n_init=10, random_state=42)
    kmeans.fit(X_scaled)
    score = silhouette_score(X_scaled, kmeans.labels_)
    silhouette_scores.append(score)

plt.plot(range(2, 11), silhouette_scores, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Analysis')
plt.show()
```




```
silhouette_scores = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, init='k-means++', n_init=10, random_state=42)
    kmeans.fit(book_df_new2)
    score = silhouette_score(book_df_new2, kmeans.labels_)
    silhouette_scores.append(score)

plt.plot(range(2, 11), silhouette_scores, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Analysis')
plt.show()
```



the inertia is very high and in this set cluster of 3 gives the accuracy of 20 % the elbow method suggested 2

```
#Find the optimal number of clusters
optimal_num_clusters = np.argmax(silhouette_scores) + 2  # Adding 2 because we started from 2 clusters
print("Optimal number of clusters:", optimal_num_clusters)
```

```
Optimal number of clusters: 2
```