Importing the Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from wordcloud import WordCloud
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import re
import warnings
warnings.filterwarnings("ignore")

nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

Download the data set

```
# Example file: google_play_reviews.csv OR netflix_reviews.csv
df = pd.read_csv("reviews.csv")  # replace with your actual file
df.head()
```

	review
0	Amazing app, very user-friendly and intuitive ...
1	Crashes every time I open it. Very frustrating.
2	I love the new update, it's smooth and fast!
3	Too many ads. It ruins the experience.
4	Great for keeping track of my tasks. Highly re...

Data Cleaning

```
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def clean_text(text):
    text = str(text).lower()
    text = re.sub(r'[^a-zA-Z]', ' ', text)  # Remove special characters
    tokens = text.split()
    tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in stop_words and len(word) > 2]
    return ' '.join(tokens)

df['cleaned_review'] = df['review'].apply(clean_text)
df['cleaned_review'].head()
```

	cleaned_review
0	amazing app user friendly intuitive interface
1	crash every time open frustrating
2	love new update smooth fast
3	many ad ruin experience
4	great keeping track task highly recommend

dtype: object

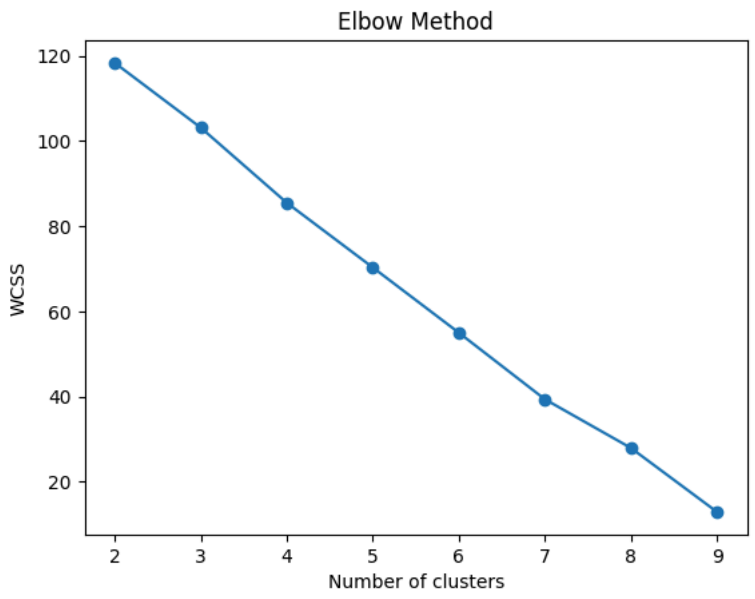
convert Text to vector

```
tfidf = TfidfVectorizer(max_features=1000)
X = tfidf.fit_transform(df['cleaned_review'])
```

Elbow Method

```
wcss = []
for i in range(2, 10):
    km = KMeans(n_clusters=i, random_state=42)
    km.fit(X)
    wcss.append(km.inertia_)

plt.plot(range(2, 10), wcss, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



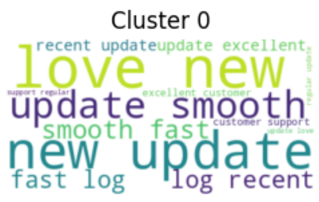
Applying Kmeans Clustering

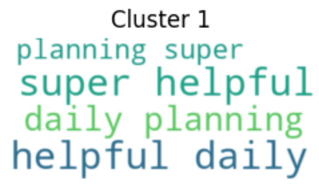
```
k = 4  # Set based on elbow method
kmeans = KMeans(n_clusters=k, random_state=42)
kmeans.fit(X)
df['cluster'] = kmeans.labels_
```

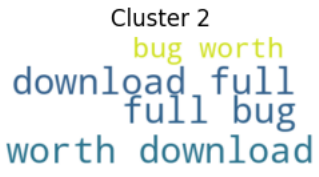
Visualize word cloud for each cluster

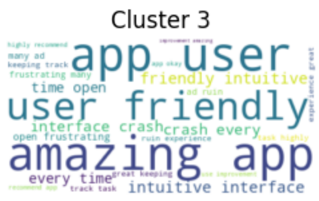
```
for i in range(k):
    cluster_text = " ".join(df[df['cluster'] == i]['cleaned_review'])
    wordcloud = WordCloud(background_color='white', max_words=100).generate(cluster_text)

    plt.figure(figsize=(3, 3))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.title(f'Cluster {i}')
    plt.axis('off')
    plt.show()
```









Analysing the Clusters

```
terms = tfidf.get_feature_names_out()
order_centroids = kmeans.cluster_centers_.argsort()[:, ::-1]

for i in range(k):
    print(f"\nCluster {i} keywords:")
    for ind in order_centroids[i, :10]:
        print(terms[ind], end=' ')
```

```
Cluster 0 keywords:
update recent log regular new fast customer excellent smooth love
Cluster 1 keywords:
super daily helpful planning update track task time support worth
Cluster 2 keywords:
worth full bug download track time task support super user
Cluster 3 keywords:
app ruin many experience ad improvement okay could use open
```