Name Maham Class BSSE 7th A. Subject NLP By Sir Abdul Ahad

```
import pandas as pd


data = pd.read_csv('/content/user_behavior_dataset.csv')

data.head()
```

	User ID	Device Model	Operating System	App Usage Time (min/day)	Screen On Time (hours/day)	Battery Drain (mAh/day)	Number of Apps Installed	(M
0	1	Google Pixel 5	Android	393	6.4	1872	67	
1	2	OnePlus 9	Android	268	4.7	1331	42	
2	3	Xiaomi Mi 11	Android	154	4.0	761	32	
3	4	Google Pixel 5	Android	239	4.8	1676	56	
4	5	iPhone 12	iOS	187	4.3	1367	58	

Next steps: Generate code with data | View recommended plots | New interactive sheet

user_behavior_ •••

1 to 10 of 700 entries Filter

User ID	Device Mo
1	Google Pix 5
2	OnePlus 9
3	Xiaomi Mi
4	Google Pix 5
5	iPhone 12
6	Google Pix 5
7	Samsung Galaxy S2
8	OnePlus 9
9	Samsung Galaxy S2
10	iPhone 12

Show 10 per page

1 2 10 60 70

Text Cleaning Now:

```
import re
import nltk
from nltk.corpus import stopwords

nltk.download('stopwords')
stop_words = set(stopwords.words('english'))

def clean_text(text):
    # Remove punctuation and special characters
    text = re.sub(r'[^A-Za-z\s]', '', text)
    # Convert to lowercase
    text = text.lower()
    # Remove stopwords
    words = [word for word in text.split() if word not in stop_words]
    return ' '.join(words)

# Apply cleaning function to your text column (replace 'text_column' with the ac
data['cleaned_text'] = data['text_column'].apply(clean_text)
```

```
print(data[['text_column', 'cleaned_text']].head())
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in
get_loc(self, key)
   3790         try:
-> 3791             return self._engine.get_loc(casted_key)
   3792         except KeyError as err:

index.pyx in pandas._libs.index.IndexEngine.get_loc()

index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'text_column'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
                                  2 frames
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in
get_loc(self, key)
   3796             ):
   3797                 raise InvalidIndexError(key)
-> 3798             raise KeyError(key) from err
   3799         except TypeError:
```

Next steps: Explain error

Stemming or Lemmatization

```
from nltk.stem import WordNetLemmatizer

nltk.download('wordnet')
lemmatizer = WordNetLemmatizer()

def lemmatize_text(text):
    words = text.split()
    return ' '.join([lemmatizer.lemmatize(word) for word in words])

data['lemmatized_text'] = data['cleaned_text'].apply(lemmatize_text)
print(data[['cleaned_text', 'lemmatized_text']].head())
```

Word Embedding

```
from gensim.models import Word2Vec

# Prepare data for Word2Vec by tokenizing sentences
sentences = [row.split() for row in data['lemmatized_text']]

# Train the Word2Vec model
model = Word2Vec(sentences, vector_size=100, window=5, min_count=1, workers=4)

# Example: get the vector for a word
word_vector = model.wv['example']  # Replace 'example' with any word from your d
print(word_vector)
```

Encoding Techniques:Bag of Words

```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer()
bow_matrix = vectorizer.fit_transform(data['lemmatized_text'])
print(bow_matrix.toarray())  # Display the Bag of Words matrix
```

One-Hot Encoding

```
from sklearn.preprocessing import OneHotEncoder

encoder = OneHotEncoder()
one_hot = encoder.fit_transform(data[['lemmatized_text']])
print(one_hot.toarray())  # Display the One-Hot Encoding matrix
```

Parts of Speech (POS) Tagging

```
nltk.download('averaged_perceptron_tagger')

# POS tagging function
def pos_tagging(text):
    words = text.split()
    return nltk.pos_tag(words)

data['pos_tags'] = data['lemmatized_text'].apply(pos_tagging)
print(data[['lemmatized_text', 'pos_tags']].head())
```

Sentiment Analysis using TextBlob:

```
from textblob import TextBlob

def get_sentiment(text):
    blob = TextBlob(text)
    return blob.sentiment.polarity

data['sentiment'] = data['lemmatized_text'].apply(get_sentiment)
print(data[['lemmatized_text', 'sentiment']].head())
```

Named Entity Recognition (NER) using SpaCy:

```
import spacy

# Load the SpaCy model for English
nlp = spacy.load("en_core_web_sm")

# Function to perform NER
def ner(text):
    doc = nlp(text)
    return [(entity.text, entity.label_) for entity in doc.ents]

data['entities'] = data['lemmatized_text'].apply(ner)
print(data[['lemmatized_text', 'entities']].head())
```