EXPERIMENT NO -05

CODE :

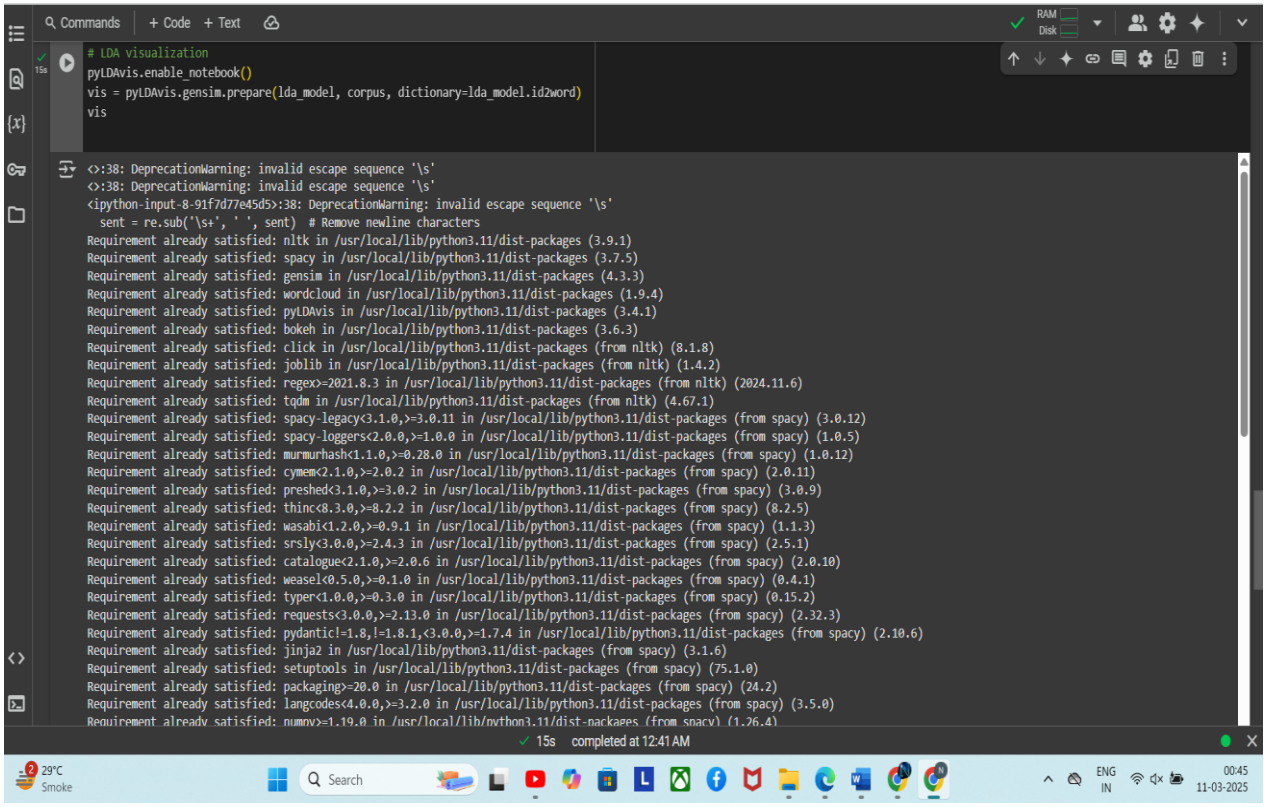
```
# Install necessary libraries
!pip install nltk spacy gensim wordcloud pyLDAvis bokeh

# Import libraries
import nltk
import re
import numpy as np
import pandas as pd
import gensim
import spacy
import logging
import warnings
import gensim.corpora as corpora
import matplotlib.pyplot as plt

from pprint import pprint
from nltk.corpus import stopwords
from gensim.utils import simple_preprocess
from gensim.models import CoherenceModel
from wordcloud import WordCloud, STOPWORDS
import matplotlib.colors as mcolors
from sklearn.manifold import TSNE
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
import pyLDAvis
import pyLDAvis.gensim

# Download stopwords
nltk.download('stopwords')
stop_words = stopwords.words('english')

# Load dataset
df = pd.read_csv('/content/google_reviews.csv')

# Preprocessing function
def sent_to_words(sentences):
    for sent in sentences:
        sent = re.sub('\s+', ' ', sent)  # Remove newline characters
        sent = re.sub("\'", "", sent)  # Remove single quotes
        sent = gensim.utils.simple_preprocess(str(sent), deacc=True)
        yield sent

# Convert to list
All_reviews = df['review_text'].values.tolist()
reviews_words = list(sent_to_words(All_reviews))
```

```
# Build bigram and trigram models
# Build bigram and trigram models
bigram = gensim.models.Phrases(reviews_words, min_count=5, threshold=10)
trigram = gensim.models.Phrases(bigram[reviews_words], threshold=10)
# Use gensim.models.phrases.Phraser directly
bigram_mod = gensim.models.phrases.Phraser(bigram)
trigram_mod = gensim.models.phrases.Phraser(trigram)

# Function for stopword removal, bigrams, trigrams, and lemmatization
def process_words(texts, stop_words=stop_words, allowed_postags=['NOUN', 'ADJ',
'VERB', 'ADV']):
    texts = [[word for word in simple_preprocess(str(doc)) if word not in stop_words] for doc
in texts]
    texts = [bigram_mod[doc] for doc in texts]
    texts = [trigram_mod[bigram_mod[doc]] for doc in texts]

    nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])
    texts_out = []

    for sent in texts:
        doc = nlp(" ".join(sent))
        texts_out.append([token.lemma_ for token in doc if token.pos_ in allowed_postags])

    # Remove stopwords again after lemmatization
    texts_out = [[word for word in simple_preprocess(str(doc)) if word not in stop_words] for
doc in texts_out]
    return texts_out

# Processed reviews
data_final = process_words(reviews_words)

# Create Dictionary and Corpus
id2word = corpora.Dictionary(data_final)
corpus = [id2word.doc2bow(text) for text in data_final]

# Build LDA Model
lda_model = gensim.models.LdaModel(
    corpus=corpus,
    id2word=id2word,
    num_topics=7,
    random_state=100,
    update_every=1,
    chunksize=10,
    passes=10,
    alpha='symmetric',
    iterations=100,
    per_word_topics=True
)

# Print the topics
```

```
pprint(lda_model.print_topics())

# Generate Word Cloud
cols = [color for name, color in mcolors.TABLEAU_COLORS.items()]
cloud = WordCloud(
    stopwords=stop_words,
    background_color='white',
    width=2500,
    height=1800,
    max_words=10,
    colormap='tab10'
)

topics = lda_model.show_topics(formatted=False)
fig, axes = plt.subplots(3, 2, figsize=(10, 10), sharex=True, sharey=True)

for i, ax in enumerate(axes.flatten()):
    fig.add_subplot(ax)
    topic_words = dict(topics[i][1])
    cloud.generate_from_frequencies(topic_words, max_font_size=300)
    plt.gca().imshow(cloud)
    plt.gca().set_title('Topic ' + str(i), fontdict=dict(size=16))
    plt.gca().axis('off')

plt.subplots_adjust(wspace=0, hspace=0)
plt.axis('off')
plt.margins(x=0, y=0)
plt.tight_layout()
plt.show()

# Get topic weights
topic_weights = []
for i, row_list in enumerate(lda_model[corpus]):
    topic_weights.append([w for i, w in row_list[0]])

# Convert to numpy array
arr = pd.DataFrame(topic_weights).fillna(0).values

# Keep the well-separated points (optional)
arr = arr[np.amax(arr, axis=1) > 0.35]

# Dominant topic number in each doc
topic_num = np.argmax(arr, axis=1)

# t-SNE Dimension Reduction
# t-SNE Dimension Reduction
tsne_model = TSNE(n_components=2, verbose=1, random_state=0, angle=.99, init='pca',
perplexity=2) # Set perplexity less than n_samples
tsne_lda = tsne_model.fit_transform(arr)
```

```
# Plot topic clusters using Bokeh
output_notebook()
n_topics = 7
mycolors = np.array([color for name, color in mcolors.TABLEAU_COLORS.items()])

plot = figure(title="t-SNE Clustering of {} LDA Topics".format(n_topics), width=900,
height=700) # Changed plot_width to width, plot_height to height
plot.scatter(x=tsne_lda[:, 0], y=tsne_lda[:, 1], color=mycolors[topic_num])
show(plot)

# LDA visualization
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, corpus, dictionary=lda_model.id2word)
vis
```

OUTPUT :



```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[(0,
  '0.086*"service" + 0.086*"buy" + 0.086*"never" + 0.086*"delivery" + '
  '0.086*"experience" + 0.086*"fast" + 0.086*"terrible" + 0.086*"great" + '
  '0.086*"average" + 0.086*"special"'),
 (1,
  '0.216*"love" + 0.216*"amazing" + 0.216*"product" + 0.027*"bad" + '
  '0.027*"well" + 0.027*"great" + 0.027*"quality" + 0.027*"special" + '
  '0.027*"terrible" + 0.027*"average"'),
 (2,
  '0.063*"bad" + 0.063*"quality" + 0.063*"average" + 0.063*"special" + '
  '0.063*"well" + 0.063*"product" + 0.063*"terrible" + 0.063*"fast" + '
  '0.062*"buy" + 0.062*"amazing"'),
 (3,
  '0.063*"product" + 0.063*"bad" + 0.063*"quality" + 0.063*"well" + '
  '0.063*"great" + 0.063*"special" + 0.063*"average" + 0.063*"fast" + '
  '0.062*"buy" + 0.062*"amazing"'),
 (4,
  '0.063*"bad" + 0.063*"quality" + 0.063*"special" + 0.063*"amazing" + '
  '0.063*"average" + 0.063*"experience" + 0.063*"well" + 0.062*"great" + '
  '0.062*"delivery" + 0.062*"love"'),
 (5,
  '0.267*"well" + 0.267*"bad" + 0.033*"special" + 0.033*"quality" + '
  '0.033*"amazing" + 0.033*"product" + 0.033*"great" + 0.033*"fast" + '
  '0.033*"terrible" + 0.033*"love"'),
 (6,
  '0.063*"bad" + 0.063*"special" + 0.063*"quality" + 0.063*"average" + '
  '0.063*"well" + 0.062*"product" + 0.062*"amazing" + 0.062*"great" + '
```

```
[t-SNE] Computing 4 nearest neighbors...
[t-SNE] Indexed 5 samples in 0.001s...
[t-SNE] Computed neighbors for 5 samples in 0.001s...
[t-SNE] Computed conditional probabilities for sample 5 / 5
[t-SNE] Mean sigma: 0.127215
[t-SNE] KL divergence after 250 iterations with early exaggeration: 71.987167
[t-SNE] KL divergence after 1000 iterations: 0.053072
```