```
import pandas as pd
import nltk
import re
import spacy
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from summa import summarizer
import warnings
from multiprocessing import Pool
from datasets import load_dataset

warnings.filterwarnings("ignore")

# Download necessary NLP models
nltk.download('stopwords')
nlp = spacy.load("en_core_web_sm")

# Load a balanced dataset (IMDB Movie Reviews)
dataset = load_dataset("imdb")
df = pd.DataFrame(dataset['train']).sample(2000, random_state=42)  # Increased sample size for better insights
df = df.rename(columns={'text': 'review', 'label': 'sentiment'})  # Ensure uniform column names

def map_sentiment(label):
    return "Positive" if label == 1 else "Negative"
df['sentiment'] = df['sentiment'].apply(map_sentiment)

# Text Preprocessing
def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)  # Remove special characters
    text = " ".join([word.lemma_ for word in nlp(text) if word.text not in nltk.corpus.stopwords.words('english')])
    return text

df['cleaned_review'] = df['review'].apply(preprocess_text)

# Topic Modeling using LDA (Faster than BERTopic)
vectorizer = CountVectorizer(stop_words="english", max_features=1000)
X = vectorizer.fit_transform(df['cleaned_review'])

lda = LatentDirichletAllocation(n_components=5, random_state=42)
df['topic'] = lda.fit_transform(X).argmax(axis=1)

# Generate Improvement Suggestions
def generate_suggestions(negative_reviews):
    text = " ".join(negative_reviews)
    if len(text.split()) < 50:  # If too short, return raw feedback
        return "Customers have reported issues such as: " + "; ".join(negative_reviews[:5])
    summary = summarizer.summarize(text, ratio=0.2)
    return summary if summary else "Common concerns include: " + "; ".join(negative_reviews[:5])

negative_reviews = df[df['sentiment'] == 'Negative']['cleaned_review'].tolist()
suggestions = generate_suggestions(negative_reviews) if negative_reviews else "No significant negative feedback found. Keep monitoring for fu

# Display Results
print("Customer Feedback Analysis Complete!\n")
print("Sample Reviews with Sentiment:")
print(df[['review', 'sentiment']].head())
print("\nMost Discussed Topics:")
print(df[['topic']].value_counts().head())
print("\nImprovement Suggestions:")
print(suggestions)

# Save results to CSV
df.to_csv("customer_feedback_analysis.csv", index=False)
```

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

README.md: 100% 7.81k/7.81k [00:00<00:00, 143kB/s]

train-00000-of-00001.parquet: 100% 21.0M/21.0M [00:00<00:00, 31.4MB/s]

test-00000-of-00001.parquet: 100% 20.5M/20.5M [00:00<00:00, 43.8MB/s]

unsupervised-00000-of-00001.parquet: 100% 42.0M/42.0M [00:00<00:00, 79.2MB/s]

Generating train split: 100% 25000/25000 [00:00<00:00, 48239.86 examples/s]

Generating test split: 100% 25000/25000 [00:00<00:00, 39544.43 examples/s]

Generating unsupervised split: 100% 50000/50000 [00:00<00:00, 69535.26 examples/s]

```
Customer Feedback Analysis Complete!

Sample Reviews with Sentiment:
                                                  review sentiment
6868   Dumb is as dumb does, in this thoroughly unint...  Negative
24016  I dug out from my garage some old musicals and...  Positive
9668   After watching this movie I was honestly disap...  Negative
13640  This movie was nominated for best picture but ...  Positive
14018  Just like Al Gore shook us up with his painful...  Positive

Most Discussed Topics:
topic
4        581
0        506
2        416
3        298
1        199
Name: count, dtype: int64

Improvement Suggestions:
Common concerns include: dumb dumb thoroughly unintereste suppose black comedy essentially start chris klein try maintain low profile ev
```