```
import pandas as pd

# Load the CSV file
df = pd.read_csv("trustpilotFlipkartReviews.csv")

# Dropping rows where at least one column is missing
df_cleaned = df.dropna(subset=["Title", "Content", "Rating"])


print(df_cleaned.head())
```

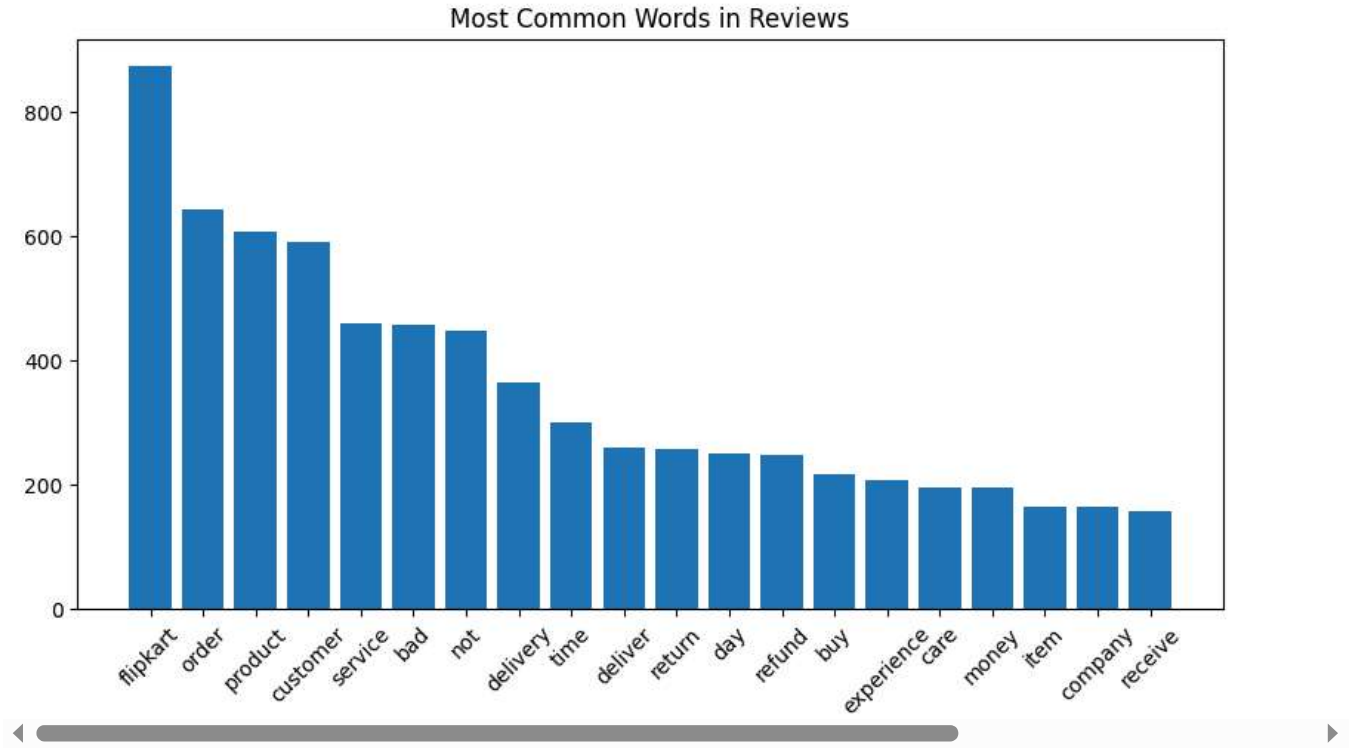
```
                                    Title  \
0      good experienxe even in electronics…
1            Rude nd arrogant delivery boys
2  They actually fall asleep when it comes…
3                   Worst shopping app ever
4       Flipkart pay later bill payment of…

                                         Content  Rating
0          good experienxe even in electronics items     5.0
1  I would never ever purchase from flip kart …it...     1.0
2  They actually fall asleep when it comes to del...     1.0
3  Worst shopping app ever. Neither do they deliv...     1.0
4  Flipkart pay later bill payment of 13767 faile...     1.0
```

Start coding or generate with AI.

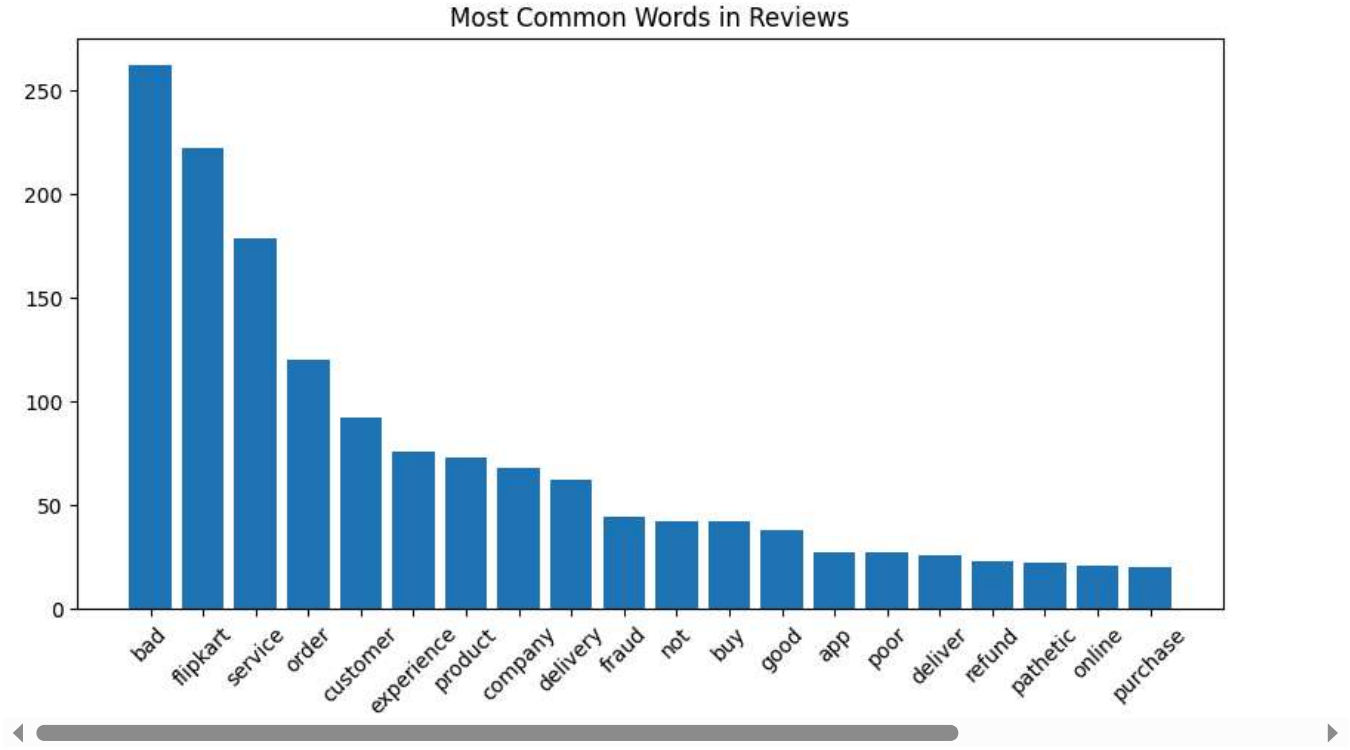
```
import spacy
import re

nlp = spacy.load("en_core_web_sm")

def clean_text(text):
    text = text.lower()  # Convert to lowercase
    text = re.sub(r"http\S+|www\S+", "", text)  # Remove URLs
    text = re.sub(r"[^\w\s]", "", text)  # Remove punctuation
    doc = nlp(text)
    tokens = [token.lemma_ for token in doc if not token.is_stop]  # Lemmatization & stopword removal
    return " ".join(tokens)

df_cleaned["Content"] = df_cleaned["Content"].apply(clean_text)
df_cleaned["Title"] = df_cleaned["Title"].apply(clean_text)
```

```
<ipython-input-2-384462964217>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-
  df_cleaned["Content"] = df_cleaned["Content"].apply(clean_text)
<ipython-input-2-384462964217>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-
  df_cleaned["Title"] = df_cleaned["Title"].apply(clean_text)
```

```
from collections import Counter
import matplotlib.pyplot as plt

words = " ".join(df_cleaned["Content"]).split()
word_freq = Counter(words).most_common(20)  # Top 20 words

plt.figure(figsize=(10, 5))
plt.bar(*zip(*word_freq))
plt.xticks(rotation=45)
plt.title("Most Common Words in Reviews")
plt.show()
```

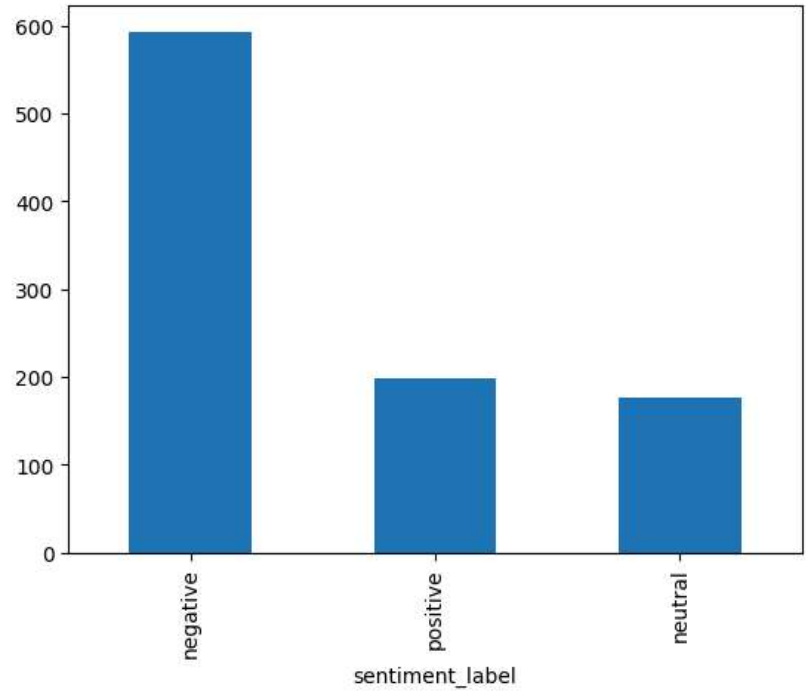



```
from collections import Counter
import matplotlib.pyplot as plt

words = " ".join(df_cleaned["Title"]).split()
word_freq = Counter(words).most_common(20)  # Top 20 words

plt.figure(figsize=(10, 5))
plt.bar(*zip(*word_freq))
plt.xticks(rotation=45)
plt.title("Most Common Words in Reviews")
plt.show()
```




```
from textblob import TextBlob

df_cleaned["sentiment"] = df_cleaned["Content"].apply(lambda x: TextBlob(x).sentiment.polarity)
df_cleaned["sentiment_label"] = df_cleaned["sentiment"].apply(lambda x: "positive" if x > 0 else ("negative" if x < 0 else '
df_cleaned["sentiment_label"].value_counts().plot(kind="bar")
```

```
<ipython-input-5-7ee89b506e7e>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-
  df_cleaned["sentiment"] = df_cleaned["Content"].apply(lambda x: TextBlob(x).sentiment.polarity)
<ipython-input-5-7ee89b506e7e>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-
  df_cleaned["sentiment_label"] = df_cleaned["sentiment"].apply(lambda x: "positive" if x > 0 else ("negative" if x <
<Axes: xlabel='sentiment_label'>
```




```
from transformers import BertTokenizer

tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
df_cleaned["tokenized_content"] = df_cleaned["Content"].apply(lambda x: tokenizer.encode(x, truncation=True, padding="max_
df_cleaned["tokenized_title"] = df_cleaned["Title"].apply(lambda x: tokenizer.encode(x, truncation=True, padding="max_leng
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 100%                                    48.0/48.0 [00:00<00:00, 1.60kB/s]
vocab.txt: 100%                                                232k/232k [00:00<00:00, 641kB/s]
tokenizer.json: 100%                                           466k/466k [00:00<00:00, 2.55MB/s]
config.json: 100%                                              570/570 [00:00<00:00, 34.8kB/s]
<ipython-input-6-7620fb00aac0>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-
  df_cleaned["tokenized_content"] = df_cleaned["Content"].apply(lambda x: tokenizer.encode(x, truncation=True, padding
<ipython-input-6-7620fb00aac0>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-
  df_cleaned["tokenized_title"] = df_cleaned["Title"].apply(lambda x: tokenizer.encode(x, truncation=True, padding="ma
```

```
print(df_cleaned.dtypes)
```

```
Title                object
Content              object
Rating              float64
sentiment           float64
sentiment_label      object
tokenized_content    object
tokenized_title      object
dtype: object
```

```
pip install pymongo
```

```
Requirement already satisfied: pymongo in /usr/local/lib/python3.11/dist-packages (4.11.3)
Requirement already satisfied: dnspython<3.0.0,>=1.16.0 in /usr/local/lib/python3.11/dist-packages (from pymongo) (2.7
```

```
from pymongo import MongoClient

# Connect to MongoDB
client = MongoClient("______")  # Replace with your MongoDB URI if using Atlas
try:
    client.admin.command('ping')
    print("Connected to MongoDB Atlas!")
except Exception as e:
    print("Error:", e)
db = client["trustpilot_reviews_db"]
collection = db["reviews"]

# Insert Data
for _, row in df_cleaned.iterrows():
    review_doc = {
        "Content": row["Content"],
        "cleaned_text": row["Title"],
        "sentiment": row["sentiment"],
        "sentiment_label": row["sentiment_label"],
        "tokenized_content": row["tokenized_content"],
        "tokenized_title": row["tokenized_title"]
    }
    collection.insert_one(review_doc)

print("Data stored successfully!")
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-10-942f445b669e> in <cell line: 0>()
----> 1 from pymongo import MongoClient
      2
      3 # Connect to MongoDB
      4 client = MongoClient("mongodb+srv://cdebodip:WsorO0MdUhMvDmGI@cluster0.mzf4aci.mongodb.net/?
retryWrites=true&w=majority&appName=Cluster0")  # Replace with your MongoDB URI if using Atlas
      5 try:

ModuleNotFoundError: No module named 'pymongo'

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
---------------------------------------------------------------------------
```

reviews = collection.find({"sentiment_label": "negative"}) # Get only negative reviews

for review in reviews: print(review["cleaned_text"])

```
client.close()
```

```
!pip install transformers torch pandas sklearn
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.11/dist-packages (4.50.0)
Requirement already satisfied: torch in /usr/local/lib/python3.11/dist-packages (2.6.0+cu124)
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Collecting sklearn
  Downloading sklearn-0.0.post12.tar.gz (2.6 kB)
  error: subprocess-exited-with-error

  × python setup.py egg_info did not run successfully.
  │ exit code: 1
  ╰─> See above for output.

  note: This error originates from a subprocess, and is likely not a problem with pip.
  Preparing metadata (setup.py) ... error
error: metadata-generation-failed

× Encountered error while generating package metadata.
╰─> See above for output.

note: This is an issue with the package mentioned above, not pip.
hint: See above for details.
```

```
# Sentiment Analysis with DistilBERT (Optimized)
from transformers import DistilBertTokenizer, DistilBertForSequenceClassification, Trainer, TrainingArguments
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
import numpy as np
import torch

# Load and split data
texts = df_cleaned["Content"].tolist()
labels = df_cleaned["sentiment_label"].tolist()
train_texts, test_texts, train_labels, test_labels = train_test_split(
    texts, labels, test_size=0.2, random_state=42
)

# Encode labels
le = LabelEncoder()
train_labels = le.fit_transform(train_labels)
test_labels = le.transform(test_labels)

# Initialize DistilBERT tokenizer
tokenizer = DistilBertTokenizer.from_pretrained("distilbert-base-uncased")

# Tokenize datasets
train_encodings = tokenizer(train_texts, truncation=True, padding=True, max_length=128)
test_encodings = tokenizer(test_texts, truncation=True, padding=True, max_length=128)

# Dataset Class
class SentimentDataset(torch.utils.data.Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

    def __getitem__(self, idx):
        item = {key: torch.tensor(val[idx]) for key, val in self.encodings.items()}
        item["labels"] = torch.tensor(self.labels[idx])
        return item

    def __len__(self):
        return len(self.labels)

# Create datasets
train_dataset = SentimentDataset(train_encodings, train_labels)
test_dataset = SentimentDataset(test_encodings, test_labels)

# Initialize DistilBERT model
model = DistilBertForSequenceClassification.from_pretrained(
    "distilbert-base-uncased",
    num_labels=len(le.classes_)
)

# Training arguments
training_args = TrainingArguments(
    output_dir="./results",
    num_train_epochs=3,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=32,
```

```
        per_device_eval_batch_size=32,
        evaluation_strategy="steps",
        eval_steps=100,
        save_steps=500,
        fp16=True,  # Enable mixed precision training
        load_best_model_at_end=True,
    )

    # Initialize Trainer
    trainer = Trainer(
        model=model,
        args=training_args,
        train_dataset=train_dataset,
        eval_dataset=test_dataset,
    )

    # Train the model
    trainer.train()

    # Evaluate
    predictions = trainer.predict(test_dataset)
    preds = np.argmax(predictions.predictions, axis=1)
    print(f"\nTest Accuracy: {accuracy_score(test_labels, preds):.2%}")

    # Prediction function
    def predict_sentiment(text):
        inputs = tokenizer(text, return_tensors="pt", truncation=True)
        with torch.no_grad():
            outputs = model(**inputs)
        return le.inverse_transform([torch.argmax(outputs.logits)])[0]

    # Test prediction
    sample_text = "This product works great and exceeded my expectations!"
    print(f"\nPrediction for '{sample_text}': {predict_sentiment(sample_text)}")
```

```
Some weights of DistilBertForSequenceClassification were not initialized from the model checkpoint at distilbert-base-
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
/usr/local/lib/python3.11/dist-packages/transformers/training_args.py:1611: FutureWarning: `evaluation_strategy` is de
  warnings.warn(
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter:
---------------------------------------------------------------------------
Abort                                     Traceback (most recent call last)
<ipython-input-11-afef1e4dbc99> in <cell line: 0>()
     72
     73 # Train the model
---> 74 trainer.train()
     75
     76 # Evaluate

                                 15 frames
/usr/local/lib/python3.11/dist-packages/click/termui.py in prompt_func(text)
    145             if hide_input:
    146                 echo(None, err=err)
--> 147             raise Abort() from None
    148
    149     if value_proc is None:

Abort:
```

```
from transformers import DistilBertTokenizer, DistilBertForSequenceClassification, Trainer, TrainingArguments
import torch
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder
import numpy as np

# 1. Data Preparation ----------------------------------------------
# Sample reduction for faster testing (remove in production)
df = df_cleaned.sample(2000, random_state=42) if len(df_cleaned) > 5000 else df_cleaned

# Label encoding
```

```
le = LabelEncoder()
df['encoded_labels'] = le.fit_transform(df['sentiment_label'])

# Train-test split
train_texts, test_texts, train_labels, test_labels = train_test_split(
    df['Content'].tolist(),
    df['encoded_labels'].tolist(),
    test_size=0.2,
    random_state=42
)

# 2. Tokenization ------------------------------------------------
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')

# Tokenize in batches (faster)
def batch_tokenize(texts, batch_size=32):
    return [tokenizer(text, truncation=True, padding=True, max_length=64) for text in texts]  # Reduced max_length

train_encodings = batch_tokenize(train_texts)
test_encodings = batch_tokenize(test_texts)

# 3. Dataset Class -----------------------------------------------
class SentimentDataset(torch.utils.data.Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

    def __getitem__(self, idx):
        item = {key: torch.tensor(val[idx]) for key, val in self.encodings.items()}
        item['labels'] = torch.tensor(self.labels[idx])
        return item

    def __len__(self):
        return len(self.labels)

train_dataset = SentimentDataset(train_encodings, train_labels)
test_dataset = SentimentDataset(test_encodings, test_labels)

# 4. Model Setup -------------------------------------------------
model = DistilBertForSequenceClassification.from_pretrained(
    'distilbert-base-uncased',
    num_labels=len(le.classes_)
).to('cuda' if torch.cuda.is_available() else 'cpu')

# 5. Optimized Training ------------------------------------------
training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=2,                    # Reduced epochs
    per_device_train_batch_size=32,        # Increased batch size
    per_device_eval_batch_size=64,
    evaluation_strategy='steps',
    eval_steps=100,
    save_steps=500,
    fp16=True,                             # Mixed precision
    load_best_model_at_end=True,
    logging_dir='./logs',
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
)

# 6. Training & Evaluation ---------------------------------------
print("Starting training...")
trainer.train()

# Predictions
predictions = trainer.predict(test_dataset)
preds = np.argmax(predictions.predictions, axis=-1)

# Decode labels
```

```
test_labels_decoded = le.inverse_transform(test_labels)
preds_decoded = le.inverse_transform(preds)

print("\nClassification Report:")
print(classification_report(test_labels_decoded, preds_decoded))

# 7. Prediction Function ----------------------------------------
def predict_sentiment(text):
    inputs = tokenizer(text, return_tensors="pt", truncation=True, max_length=64).to(model.device)
    with torch.no_grad():
        outputs = model(**inputs)
    return le.inverse_transform([torch.argmax(outputs.logits).item()])[0]

# Test Cases
test_phrases = [
    "This product works perfectly!",
    "Worst customer service ever",
    "It's okay, not great"
]

for phrase in test_phrases:
    print(f"\nText: {phrase}")
    print(f"Prediction: {predict_sentiment(phrase)}")
```

```
<ipython-input-11-9be0e0ca2251>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-
  df['encoded_labels'] = le.fit_transform(df['sentiment_label'])
tokenizer_config.json: 100%                                  48.0/48.0 [00:00<00:00, 4.07kB/s]
vocab.txt: 100%                                              232k/232k [00:00<00:00, 12.8MB/s]
tokenizer.json: 100%                                         466k/466k [00:00<00:00, 19.0MB/s]
config.json: 100%                                            483/483 [00:00<00:00, 48.9kB/s]
model.safetensors: 100%                                      268M/268M [00:02<00:00, 133MB/s]
Some weights of DistilBertForSequenceClassification were not initialized from the model checkpoint at distilbert-base-
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
/usr/local/lib/python3.11/dist-packages/transformers/training_args.py:1611: FutureWarning: `evaluation_strategy` is de
  warnings.warn(
wandb: WARNING The `run_name` is currently set to the same value as `TrainingArguments.output_dir`. If this was not in
Starting training...
wandb: Using wandb-core as the SDK backend.  Please refer to https://wandb.me/wandb-core for more information.
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter:
---------------------------------------------------------------------------
Abort                                     Traceback (most recent call last)
<ipython-input-11-9be0e0ca2251> in <cell line: 0>()
     78 # 6. Training & Evaluation ----------------------------------------
     79 print("Starting training...")
---> 80 trainer.train()
     81
     82 # Predictions

                                 15 frames
/usr/local/lib/python3.11/dist-packages/click/termui.py in prompt_func(text)
    145             if hide_input:
    146                 echo(None, err=err)
--> 147             raise Abort() from None
    148
    149     if value_proc is None:

Abort:
```

```
import pandas as pd
import fasttext
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

# 1. Prepare Data ----------------------------------------------------
```

```
# Convert to FastText format (__label__prefix required)
df_cleaned['fasttext_format'] = df_cleaned.apply(
    lambda row: f"__label__{row['sentiment_label']} {row['Content']}",
    axis=1
)

# Split data (80% train, 20% test)
train, test = train_test_split(df_cleaned, test_size=0.2, random_state=42)

# Save to text files (FastText requires file input)
train[['fasttext_format']].to_csv('fasttext_train.txt',
                                  index=False,
                                  header=False,
                                  sep='\n')
test[['fasttext_format']].to_csv('fasttext_test.txt',
                                 index=False,
                                 header=False,
                                 sep='\n')

# 2. Train Model ----------------------------------------------------
model = fasttext.train_supervised(
    input='fasttext_train.txt',
    epoch=50,            # Number of iterations
    lr=0.5,              # Learning rate
    wordNgrams=2,        # Uses word pairs (bigrams)
    dim=100,             # Embedding dimension
    loss='ova'           # One-vs-all for multi-class
)

# 3. Evaluate -------------------------------------------------------
def fasttext_predict(text):
    # Convert text to list before prediction
    labels, probs = model.predict([text.replace('\n', ' ')], k=1)
    # Return the first predicted label, removing '__label__' prefix
    return labels[0][0].replace('__label__', '')

# Get predictions
test['predicted_label'] = test['Content'].apply(fasttext_predict)

# Classification report
print(classification_report(test['sentiment_label'],
                           test['predicted_label']))

# 4. Save & Load Model ----------------------------------------------
model.save_model('sentiment_model.ftz')

# Later load with:
# model = fasttext.load_model('sentiment_model.ftz')

# 5. Make Predictions -----------------------------------------------
sample_texts = [
    "This product works perfectly!",
    "Worst customer service ever",
    "It's okay, not great"
]

for text in sample_texts:
    print(f"Text: {text}")
    print(f"Prediction: {fasttext_predict(text)}\n")
```

```
<ipython-input-10-e1128c589afc>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-
  df_cleaned['fasttext_format'] = df_cleaned.apply(
              precision    recall  f1-score   support

    negative       0.76      0.94      0.84       123
     neutral       0.70      0.55      0.62        29
    positive       0.58      0.26      0.36        42

    accuracy                           0.74       194
   macro avg       0.68      0.59      0.61       194
weighted avg       0.71      0.74      0.70       194
```

```
Text: This product works perfectly!
Prediction: neutral

Text: Worst customer service ever
Prediction: positive

Text: It's okay, not great
Prediction: neutral
```

```
!pip install fasttext
```

```
Collecting fasttext
  Downloading fasttext-0.9.3.tar.gz (73 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 73.4/73.4 kB 2.0 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting pybind11>=2.2 (from fasttext)
  Using cached pybind11-2.13.6-py3-none-any.whl.metadata (9.5 kB)
Requirement already satisfied: setuptools>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from fasttext) (75.2.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from fasttext) (2.0.2)
Using cached pybind11-2.13.6-py3-none-any.whl (243 kB)
Building wheels for collected packages: fasttext
  Building wheel for fasttext (pyproject.toml) ... done
  Created wheel for fasttext: filename=fasttext-0.9.3-cp311-cp311-linux_x86_64.whl size=4313505 sha256=c5ef25b3a9e7e48
  Stored in directory: /root/.cache/pip/wheels/65/4f/35/5057db0249224e9ab55a513fa6b79451473ceb7713017823c3
Successfully built fasttext
Installing collected packages: pybind11, fasttext
Successfully installed fasttext-0.9.3 pybind11-2.13.6
```

```
import pandas as pd
import fasttext
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

# 1. DATA PREPARATION ---------------------------------------------
# Clean text (critical for FastText)
def clean_text(text):
    text = str(text).lower()
    text = text.replace('\n', ' ').replace('\r', '')
    return ' '.join(text.split())

df_cleaned['cleaned_content'] = df_cleaned['Content'].apply(clean_text)

# Balance classes (undersample negative)
negative_samples = df_cleaned[df_cleaned['sentiment_label'] == 'negative']
neutral_samples = df_cleaned[df_cleaned['sentiment_label'] == 'neutral']
positive_samples = df_cleaned[df_cleaned['sentiment_label'] == 'positive']

# Take min samples per class (or use oversampling)
min_samples = min(len(negative_samples), len(neutral_samples), len(positive_samples))
balanced_df = pd.concat([
    negative_samples.sample(min_samples, random_state=42),
    neutral_samples.sample(min_samples, random_state=42),
    positive_samples.sample(min_samples, random_state=42)
])

# 2. MODEL TRAINING -----------------------------------------------
# Format for FastText
balanced_df['fasttext_format'] = balanced_df.apply(
    lambda row: f"__label__{row['sentiment_label']} {row['cleaned_content']}",
    axis=1
)

# Split data
train, test = train_test_split(balanced_df, test_size=0.2, random_state=42)

# Save files
train[['fasttext_format']].to_csv('balanced_train.txt', index=False, header=False, sep='\n')
test[['fasttext_format']].to_csv('balanced_test.txt', index=False, header=False, sep='\n')

# Hyperparameter tuning
```

```
model = fasttext.train_supervised(
    input='balanced_train.txt',
    epoch=100,            # More iterations
    lr=0.05,             # Lower learning rate
    wordNgrams=3,        # Better context capture
    loss='ova',          # One-vs-all for multi-class
    minCount=2,          # Ignore rare words
    bucket=200000        # More hash buckets
)

# 3. EVALUATION ----------------------------------------------------
def predict_sentiment(text):
    text = clean_text(text)
    labels, probs = model.predict([text.replace('\n', ' ')], k=1)
    return labels[0][0].replace('__label__', '')

test['prediction'] = test['cleaned_content'].apply(predict_sentiment)
print(classification_report(test['sentiment_label'], test['prediction']))

# 4. TEST CASES ----------------------------------------------------
test_phrases = [
    "This product works perfectly!",
    "Worst customer service ever",
    "It's okay, not great",
    "Absolutely fantastic experience",
    "The quality is mediocre at best"
]

for phrase in test_phrases:
    print(f"Text: {phrase}")
    print(f"Prediction: {predict_sentiment(phrase)}\n")
```

```
<ipython-input-16-13c78da6a990>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-
  df_cleaned['cleaned_content'] = df_cleaned['Content'].apply(clean_text)
              precision    recall  f1-score   support

    negative       0.68      0.61      0.64        46
     neutral       0.48      0.64      0.55        25
    positive       0.61      0.56      0.59        39

    accuracy                           0.60       110
   macro avg       0.59      0.60      0.59       110
weighted avg       0.61      0.60      0.60       110

Text: This product works perfectly!
Prediction: neutral

Text: Worst customer service ever
Prediction: negative

Text: It's okay, not great
Prediction: neutral

Text: Absolutely fantastic experience
Prediction: neutral

Text: The quality is mediocre at best
Prediction: neutral
```

```
pip install lightgbm
```

```
Requirement already satisfied: lightgbm in /usr/local/lib/python3.11/dist-packages (4.5.0)
Requirement already satisfied: numpy>=1.17.0 in /usr/local/lib/python3.11/dist-packages (from lightgbm) (2.0.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from lightgbm) (1.14.1)
```

```
from lightgbm import LGBMClassifier
from sentence_transformers import SentenceTransformer
from sklearn.utils import compute_class_weight
import numpy as np
model = SentenceTransformer('all-mpnet-base-v2')  # ~90MB, very fast
```

```
X_train = model.encode(train['Content'].tolist(), show_progress_bar=True)
X_test = model.encode(test['Content'].tolist(), show_progress_bar=True)
class_weights = compute_class_weight('balanced', classes=np.unique(train['sentiment_label']), y=train['sentiment_label'])
class_weight_dict = dict(zip(np.unique(train['sentiment_label']), class_weights))
lgb = LGBMClassifier(class_weight=class_weight_dict, random_state=42)  # Same as before
lgb.fit(X_train, train['sentiment_label'])
test['lgb_pred'] = lgb.predict(X_test)
print(classification_report(test['sentiment_label'], test['lgb_pred']))
```

```
Batches: 100%                                   25/25 [02:47<00:00,  1.25s/it]

Batches: 100%                                   7/7 [01:06<00:00,  3.83s/it]
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'force_all_finite' was rename
  warnings.warn(
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.015870 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 195801
[LightGBM] [Info] Number of data points in the train set: 775, number of used features: 768
[LightGBM] [Info] Start training from score -1.098612
[LightGBM] [Info] Start training from score -1.098612
[LightGBM] [Info] Start training from score -1.098612
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
              precision    recall  f1-score   support

    negative       0.71      0.91      0.80       123
     neutral       0.76      0.45      0.57        29
    positive       0.55      0.26      0.35        42

    accuracy                           0.70       194
   macro avg       0.68      0.54      0.57       194
weighted avg       0.69      0.70      0.67       194

/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'force_all_finite' was rename
  warnings.warn(
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Disable normalization to keep all TF-IDF values non-negative
tfidf = TfidfVectorizer(max_features=5000, ngram_range=(1,2), norm=None)
X_train = tfidf.fit_transform(train['Content'])
X_test = tfidf.transform(test['Content'])
from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB()
nb.fit(X_train, train['sentiment_label'])
test['nb_pred'] = nb.predict(X_test)

from sklearn.metrics import classification_report
print(classification_report(test['sentiment_label'], test['nb_pred']))
```

```
              precision    recall  f1-score   support

    negative       0.73      0.85      0.78       123
     neutral       0.58      0.38      0.46        29
    positive       0.45      0.36      0.40        42

    accuracy                           0.67       194
   macro avg       0.59      0.53      0.55       194
weighted avg       0.65      0.67      0.65       194
```

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, train['sentiment_label'])
test['lr_pred'] = lr.predict(X_test)

print(classification_report(test['sentiment_label'], test['lr_pred']))
```

```
              precision    recall  f1-score   support

    negative       0.81      0.91      0.85       123
     neutral       0.64      0.62      0.63        29
    positive       0.67      0.43      0.52        42

    accuracy                           0.76       194
   macro avg       0.71      0.65      0.67       194
weighted avg       0.75      0.76      0.75       194
```

```
from sklearn.svm import LinearSVC
svc = LinearSVC()
svc.fit(X_train, train['sentiment_label'])
test['svc_pred'] = svc.predict(X_test)

print(classification_report(test['sentiment_label'], test['svc_pred']))
```

```
              precision    recall  f1-score   support

    negative       0.83      0.84      0.83       123
     neutral       0.46      0.62      0.53        29
    positive       0.71      0.52      0.60        42

    accuracy                           0.74       194
   macro avg       0.67      0.66      0.66       194
weighted avg       0.75      0.74      0.74       194
```

```
from sklearn.naive_bayes import GaussianNB
tfidf = TfidfVectorizer(max_features=5000, ngram_range=(1,2), norm=None)
X_train = tfidf.fit_transform(train['Content'])
X_test = tfidf.transform(test['Content'])
from sklearn.naive_bayes import MultinomialNB
nb = GaussianNB()
nb.fit(X_train, train['sentiment_label'])
test['nb_pred'] = nb.predict(X_test)

from sklearn.metrics import classification_report
print(classification_report(test['sentiment_label'], test['nb_pred']))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-27-f4b8d269071a> in <cell line: 0>()
      5 from sklearn.naive_bayes import MultinomialNB
      6 nb = GaussianNB()
----> 7 nb.fit(X_train, train['sentiment_label'])
      8 test['nb_pred'] = nb.predict(X_test)
      9

6 frames
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py in _ensure_sparse_format(sparse_container,
accept_sparse, dtype, copy, ensure_all_finite, accept_large_sparse, estimator_name, input_name)
    611     if accept_sparse is False:
    612         padded_input = " for " + input_name if input_name else ""
--> 613         raise TypeError(
    614             f"Sparse data was passed{padded_input}, but dense data is required. "
    615             "Use '.toarray()' to convert to a dense numpy array."

TypeError: Sparse data was passed for X, but dense data is required. Use '.toarray()' to convert to a dense numpy
array.
```

```
from sklearn.preprocessing import LabelEncoder

# Encode string labels to integers
```

```
le = LabelEncoder()
train_labels_encoded = le.fit_transform(train['sentiment_label'])
test_labels_encoded = le.transform(test['sentiment_label'])

# Train XGBoost with encoded labels
xgb_model.fit(X_train, train_labels_encoded)

# Predict and decode predictions
test['xgb_pred_encoded'] = xgb_model.predict(X_test)
test['xgb_pred'] = le.inverse_transform(test['xgb_pred_encoded'])

# Evaluate
from sklearn.metrics import classification_report
print(classification_report(test['sentiment_label'], test['xgb_pred']))
```

```
              precision    recall  f1-score   support

    negative       0.84      0.87      0.85       123
     neutral       0.62      0.72      0.67        29
    positive       0.69      0.52      0.59        42

    accuracy                           0.77       194
   macro avg       0.71      0.71      0.70       194
weighted avg       0.77      0.77      0.77       194
```

```
from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report

# Encode labels
le = LabelEncoder()
y_train = le.fit_transform(train['sentiment_label'])
y_test = le.transform(test['sentiment_label'])

# Base model
xgb = XGBClassifier(objective='multi:softmax', num_class=3, use_label_encoder=False, eval_metric='mlogloss', random_state=

# Grid of hyperparameters to tune
param_grid = {
    'learning_rate': [0.01, 0.1, 0.3],
    'max_depth': [4, 6, 8],
    'n_estimators': [100, 200],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}

# Grid search with cross-validation
grid_search = GridSearchCV(estimator=xgb, param_grid=param_grid,
                           cv=3, scoring='accuracy', n_jobs=-1, verbose=2)

# Fit the model
grid_search.fit(X_train, y_train)

# Get best model
best_model = grid_search.best_estimator_

# Predict and decode predictions
y_pred = best_model.predict(X_test)
y_pred_labels = le.inverse_transform(y_pred)

# Evaluation
print("Best Parameters:", grid_search.best_params_)
print(classification_report(test['sentiment_label'], y_pred_labels))
```

```
Fitting 3 folds for each of 72 candidates, totalling 216 fits
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [17:11:51] WARNING: /workspace/src/learner.c
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
Best Parameters: {'colsample_bytree': 0.8, 'learning_rate': 0.1, 'max_depth': 4, 'n_estimators': 200, 'subsample': 1.0
              precision    recall  f1-score   support

    negative       0.84      0.89      0.87       123
```

```
           neutral       0.60      0.72      0.66        29
          positive       0.67      0.48      0.56        42

          accuracy                           0.77       194
         macro avg       0.70      0.70      0.69       194
      weighted avg       0.77      0.77      0.77       194
```

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, train['sentiment_label'])
test['rf_pred'] = rf.predict(X_test)

print(classification_report(test['sentiment_label'], test['rf_pred']))
```

```
              precision    recall  f1-score   support

        negative       0.77      0.93      0.84       123
         neutral       0.57      0.59      0.58        29
        positive       0.86      0.29      0.43        42

        accuracy                           0.74       194
       macro avg       0.73      0.60      0.62       194
    weighted avg       0.76      0.74      0.71       194
```

```
pip install catboost
```

```
Collecting catboost
  Downloading catboost-1.2.8-cp311-cp311-manylinux2014_x86_64.whl.metadata (1.2 kB)
Requirement already satisfied: graphviz in /usr/local/lib/python3.11/dist-packages (from catboost) (0.20.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (from catboost) (3.10.0)
Requirement already satisfied: numpy<3.0,>=1.16.0 in /usr/local/lib/python3.11/dist-packages (from catboost) (2.0.2)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.11/dist-packages (from catboost) (2.2.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from catboost) (1.14.1)
Requirement already satisfied: plotly in /usr/local/lib/python3.11/dist-packages (from catboost) (5.24.1)
Requirement already satisfied: six in /usr/local/lib/python3.11/dist-packages (from catboost) (1.17.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas>=0.24->c
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=0.24->catboost) (
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=0.24->catboost)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->catboost)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib->catboost) (0.
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->catboost
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->catboost
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->catboost)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib->catboost) (11.1.
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->catboost)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.11/dist-packages (from plotly->catboost) (9.1
Downloading catboost-1.2.8-cp311-cp311-manylinux2014_x86_64.whl (99.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 99.2/99.2 MB 6.7 MB/s eta 0:00:00
Installing collected packages: catboost
Successfully installed catboost-1.2.8
```

```
from catboost import CatBoostClassifier
from sklearn.metrics import classification_report

# Label encoding if your labels are still strings
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y_train = le.fit_transform(train['sentiment_label'])
y_test = le.transform(test['sentiment_label'])

# Create and fit the CatBoost model
cat_model = CatBoostClassifier(
    iterations=200,
    learning_rate=0.1,
    depth=6,
    verbose=0,
    random_state=42
)

cat_model.fit(X_train, y_train)

# Make predictions
```

```
y_pred = cat_model.predict(X_test)

# Decode predictions back to labels
y_pred_labels = le.inverse_transform(y_pred)

# Print classification report
print(classification_report(test['sentiment_label'], y_pred_labels))
```

```
              precision    recall  f1-score   support

    negative       0.76      0.92      0.83       123
     neutral       0.55      0.55      0.55        29
    positive       0.71      0.29      0.41        42

    accuracy                           0.73       194
   macro avg       0.67      0.59      0.60       194
weighted avg       0.72      0.73      0.70       194

/usr/local/lib/python3.11/dist-packages/sklearn/preprocessing/_label.py:151: DataConversionWarning: A column-vector y
  y = column_or_1d(y, warn=True)
```

```
#CNN
import numpy as np
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Conv1D, GlobalMaxPooling1D, Dense, Dropout
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report

# ----------------------------
# 1. Tokenize and prepare data
# ----------------------------
vocab_size = 10000
max_len = 100
embedding_dim = 128

tokenizer = Tokenizer(num_words=vocab_size, oov_token="<OOV>")
tokenizer.fit_on_texts(train['Content'])

X_train_seq = tokenizer.texts_to_sequences(train['Content'])
X_test_seq = tokenizer.texts_to_sequences(test['Content'])

X_train_pad = pad_sequences(X_train_seq, maxlen=max_len, padding='post')
X_test_pad = pad_sequences(X_test_seq, maxlen=max_len, padding='post')

# ----------------------------
# 2. Encode labels
# ----------------------------
le = LabelEncoder()
y_train = le.fit_transform(train['sentiment_label'])
y_test = le.transform(test['sentiment_label'])

# ----------------------------
# 3. Build CNN model
# ----------------------------
model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=embedding_dim, input_length=max_len),
    Conv1D(128, kernel_size=5, activation='relu'),
    GlobalMaxPooling1D(),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dense(3, activation='softmax')  # Change to match number of classes
])

model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# ----------------------------
# 4. Train model
# ----------------------------
model.fit(X_train_pad, y_train, validation_split=0.1, epochs=5, batch_size=32)
```

```
# ----------------------------
# 5. Evaluate
# ----------------------------
y_pred = model.predict(X_test_pad)
y_pred_labels = np.argmax(y_pred, axis=1)

print(classification_report(le.inverse_transform(y_test), le.inverse_transform(y_pred_labels)))
```

```
Epoch 1/5
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is
  warnings.warn(
22/22 ━━━━━━━━━━━━━━━━━━━━ 5s 103ms/step - accuracy: 0.5781 - loss: 1.0225 - val_accuracy: 0.6154 - val_loss: 0.8974
Epoch 2/5
22/22 ━━━━━━━━━━━━━━━━━━━━ 2s 87ms/step - accuracy: 0.6094 - loss: 0.8752 - val_accuracy: 0.6154 - val_loss: 0.8051
Epoch 3/5
22/22 ━━━━━━━━━━━━━━━━━━━━ 3s 113ms/step - accuracy: 0.6374 - loss: 0.7347 - val_accuracy: 0.6667 - val_loss: 0.7067
Epoch 4/5
22/22 ━━━━━━━━━━━━━━━━━━━━ 5s 122ms/step - accuracy: 0.7600 - loss: 0.5883 - val_accuracy: 0.7308 - val_loss: 0.6347
Epoch 5/5
22/22 ━━━━━━━━━━━━━━━━━━━━ 2s 81ms/step - accuracy: 0.8823 - loss: 0.4149 - val_accuracy: 0.7949 - val_loss: 0.5262
WARNING:tensorflow:5 out of the last 15 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_o
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
              precision    recall  f1-score   support

    negative       0.78      0.92      0.85       123
     neutral       0.79      0.52      0.62        29
    positive       0.68      0.50      0.58        42

    accuracy                           0.77       194
   macro avg       0.75      0.65      0.68       194
weighted avg       0.76      0.77      0.75       194
```

```
import numpy as np
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Conv1D, GlobalMaxPooling1D, Dense, Dropout, BatchNormalization
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from scikeras.wrappers import KerasClassifier  # Updated import
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
# ----------------------------
# 1. Tokenize and prepare data
# ----------------------------
vocab_size = 10000
max_len = 100
embedding_dim = 128

tokenizer = Tokenizer(num_words=vocab_size, oov_token="<OOV>")
tokenizer.fit_on_texts(train['Content'])

X_train_seq = tokenizer.texts_to_sequences(train['Content'])
X_test_seq = tokenizer.texts_to_sequences(test['Content'])

X_train_pad = pad_sequences(X_train_seq, maxlen=max_len, padding='post')
X_test_pad = pad_sequences(X_test_seq, maxlen=max_len, padding='post')

# ----------------------------
# 2. Encode labels
# ----------------------------
le = LabelEncoder()
y_train = le.fit_transform(train['sentiment_label'])
y_test = le.transform(test['sentiment_label'])

# ----------------------------
# 3. Define model builder
# ----------------------------
def create_model(optimizer='adam', dropout_rate=0.5):
    model = Sequential([
```

```
            Embedding(input_dim=vocab_size, output_dim=embedding_dim, input_length=max_len),
            Conv1D(128, kernel_size=5, activation='relu'),
            BatchNormalization(),
            GlobalMaxPooling1D(),
            Dropout(dropout_rate),
            Dense(64, activation='relu'),
            Dropout(dropout_rate),
            Dense(3, activation='softmax')  # 3 classes
        ])
        model.compile(loss='sparse_categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
        return model

    # ----------------------------
    # 4. Setup KerasClassifier (SciKeras)
    # ----------------------------
    early_stopping = EarlyStopping(monitor='val_loss', patience=3)
    reduce_lr = ReduceLROnPlateau(monitor='val_loss', patience=2, factor=0.5, min_lr=1e-6)

    from sklearn.metrics import accuracy_score

    # Define hyperparameter combinations manually
    param_combinations = [
        {"batch_size": 16, "epochs": 5, "optimizer": "adam", "dropout_rate": 0.3},
        {"batch_size": 32, "epochs": 10, "optimizer": "sgd", "dropout_rate": 0.5},
        # Add more combinations as needed
    ]

    best_score = -1
    best_params = {}

    for params in param_combinations:
        print(f"\nTesting params: {params}")

        # Create and train model
        model = create_model(
            optimizer=params["optimizer"],
            dropout_rate=params["dropout_rate"]
        )

        model.fit(
            X_train_pad, y_train,
            batch_size=params["batch_size"],
            epochs=params["epochs"],
            validation_split=0.1,
            verbose=0
        )

        # Evaluate
        y_pred = np.argmax(model.predict(X_test_pad), axis=1)
        score = accuracy_score(y_test, y_pred)

        if score > best_score:
            best_score = score
            best_params = params

    print(f"\nBest params: {best_params}, Accuracy: {best_score:.4f}")

    #best_model = grid_result.best_estimator_
    #y_pred_proba = best_model.predict_proba(X_test_pad)
    #y_pred_labels = np.argmax(y_pred_proba, axis=1)

    #print(classification_report(y_test, y_pred_labels))
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-15-5f85cdc55301> in <cell line: 0>()
      8 from sklearn.metrics import classification_report
      9 from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
---> 10 from scikeras.wrappers import KerasClassifier  # Updated import
     11 from sklearn.model_selection import GridSearchCV
     12 from sklearn.metrics import accuracy_score

ModuleNotFoundError: No module named 'scikeras'

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
---------------------------------------------------------------------------
```

```
#final_model
#CNN with keras search
import numpy as np
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Conv1D, GlobalMaxPooling1D, Dense, Dropout
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from kerastuner import HyperModel, RandomSearch
from tensorflow.keras.callbacks import EarlyStopping

# ----------------------------
# 1. Load and Prepare Data
# ----------------------------
# Assuming you have train/test DataFrames with 'Content' and 'sentiment_label'
# If not, replace with your data loading logic:
# train = pd.read_csv('your_data.csv')

# Tokenization
vocab_size = 10000
max_len = 100
embedding_dim = 128

tokenizer = Tokenizer(num_words=vocab_size, oov_token="<OOV>")
tokenizer.fit_on_texts(train['Content'])

X_train_seq = tokenizer.texts_to_sequences(train['Content'])
X_test_seq = tokenizer.texts_to_sequences(test['Content'])

X_train_pad = pad_sequences(X_train_seq, maxlen=max_len, padding='post')
X_test_pad = pad_sequences(X_test_seq, maxlen=max_len, padding='post')

# Label Encoding
le = LabelEncoder()
y_train = le.fit_transform(train['sentiment_label'])
y_test = le.transform(test['sentiment_label'])

# ----------------------------
# 2. Define HyperModel
# ----------------------------
class SentimentHyperModel(HyperModel):
    def build(self, hp):
        model = Sequential([
            Embedding(input_dim=vocab_size,
                      output_dim=hp.Int('embedding_dim', min_value=64, max_value=256, step=64),
                      input_length=max_len),

            Conv1D(filters=hp.Int('conv_filters', min_value=64, max_value=256, step=64),
                   kernel_size=hp.Choice('kernel_size', values=[3, 5, 7]),
                   activation='relu',
                   padding='same'),

            GlobalMaxPooling1D(),
```

```
            Dense(units=hp.Int('dense_units', min_value=32, max_value=128, step=32),
                  activation='relu'),

            Dropout(rate=hp.Float('dropout_rate', min_value=0.3, max_value=0.7, step=0.1)),

            Dense(3, activation='softmax')
        ])

        optimizer = hp.Choice('optimizer', values=['adam', 'rmsprop', 'sgd'])
        model.compile(optimizer=optimizer,
                      loss='sparse_categorical_crossentropy',
                      metrics=['accuracy'])
        return model

# ----------------------------
# 3. Initialize Tuner
# ----------------------------
tuner = RandomSearch(
    SentimentHyperModel(),
    objective='val_accuracy',
    max_trials=10,  # Number of hyperparameter combinations to try
    executions_per_trial=2,  # Run each trial twice for reliability
    directory='keras_tuner',
    project_name='sentiment_analysis'
)

# Early stopping callback
early_stop = EarlyStopping(monitor='val_loss', patience=3)

# ----------------------------
# 4. Run Hyperparameter Search
# ----------------------------
tuner.search(X_train_pad, y_train,
             epochs=20,
             validation_split=0.2,
             callbacks=[early_stop],
             verbose=2)

# ----------------------------
# 5. Retrieve Best Model
# ----------------------------
# Get the top 2 models
best_models = tuner.get_best_models(num_models=2)
best_model = tuner.get_best_models(num_models=1)[0]

# Get the best hyperparameters
best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]

print(f"""
Best hyperparameters:
- Embedding dim: {best_hps.get('embedding_dim')}
- Conv filters: {best_hps.get('conv_filters')}
- Kernel size: {best_hps.get('kernel_size')}
- Dense units: {best_hps.get('dense_units')}
- Dropout rate: {best_hps.get('dropout_rate')}
- Optimizer: {best_hps.get('optimizer')}
""")

# ----------------------------
# 6. Evaluate on Test Set
# ----------------------------
# Train final model with best hyperparameters
history = best_model.fit(
    X_train_pad, y_train,
    batch_size=32,
    epochs=20,
    validation_split=0.2,
    callbacks=[early_stop],
    verbose=1
)

# Evaluate
test_loss, test_acc = best_model.evaluate(X_test_pad, y_test)
```

```
print(f"\nTest Accuracy: {test_acc:.4f}")

# Generate predictions
y_pred = np.argmax(best_model.predict(X_test_pad), axis=1)
print(classification_report(y_test, y_pred))
```

```
Reloading Tuner from keras_tuner/sentiment_analysis/tuner0.json
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/keras/src/saving/saving_lib.py:757: UserWarning: Skipping variable loading for
  saveable.load_own_variables(weights_store.get(inner_path))
/usr/local/lib/python3.11/dist-packages/keras/src/saving/saving_lib.py:757: UserWarning: Skipping variable loading for
  saveable.load_own_variables(weights_store.get(inner_path))

Best hyperparameters:
- Embedding dim: 64
- Conv filters: 128
- Kernel size: 3
- Dense units: 32
- Dropout rate: 0.4
- Optimizer: adam

Epoch 1/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 2s 38ms/step - accuracy: 0.9892 - loss: 0.0725 - val_accuracy: 0.8000 - val_loss: 0.4244
Epoch 2/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.9886 - loss: 0.0692 - val_accuracy: 0.8194 - val_loss: 0.4529
Epoch 3/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - accuracy: 0.9952 - loss: 0.0526 - val_accuracy: 0.8194 - val_loss: 0.5500
Epoch 4/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.9954 - loss: 0.0273 - val_accuracy: 0.8194 - val_loss: 0.5653
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8024 - loss: 0.6217

Test Accuracy: 0.7990
WARNING:tensorflow:5 out of the last 15 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_o
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
              precision    recall  f1-score   support

           0       0.82      0.93      0.87       123
           1       0.77      0.59      0.67        29
           2       0.72      0.55      0.62        42

    accuracy                           0.80       194
   macro avg       0.77      0.69      0.72       194
weighted avg       0.79      0.80      0.79       194
```

```
!pip install keras-tuner
```

```
Collecting keras-tuner
  Downloading keras_tuner-1.4.7-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: keras in /usr/local/lib/python3.11/dist-packages (from keras-tuner) (3.8.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from keras-tuner) (24.2)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from keras-tuner) (2.32.3)
Collecting kt-legacy (from keras-tuner)
  Downloading kt_legacy-1.0.5-py3-none-any.whl.metadata (221 bytes)
Requirement already satisfied: absl-py in /usr/local/lib/python3.11/dist-packages (from keras->keras-tuner) (1.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from keras->keras-tuner) (2.0.2)
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from keras->keras-tuner) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (from keras->keras-tuner) (0.0.8)
Requirement already satisfied: h5py in /usr/local/lib/python3.11/dist-packages (from keras->keras-tuner) (3.13.0)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (from keras->keras-tuner) (0.15.0)
Requirement already satisfied: ml-dtypes in /usr/local/lib/python3.11/dist-packages (from keras->keras-tuner) (0.4.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->ker
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->keras-tuner) (3
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->keras-tun
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->keras-tun
Requirement already satisfied: typing-extensions>=4.5.0 in /usr/local/lib/python3.11/dist-packages (from optree->keras
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras->ker
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras->k
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich
Downloading keras_tuner-1.4.7-py3-none-any.whl (129 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 129.1/129.1 kB 2.4 MB/s eta 0:00:00
Downloading kt_legacy-1.0.5-py3-none-any.whl (9.6 kB)
Installing collected packages: kt-legacy, keras-tuner
Successfully installed keras-tuner-1.4.7 kt-legacy-1.0.5
```

```
import pickle
from tensorflow.keras.models import save_model

# Save the trained model
save_model(best_model, 'flipkart_sentiment_model.keras')  # or .h5

# Save the tokenizer (critical for text preprocessing)
with open('tokenizer.pkl', 'wb') as handle:
    pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST_PROTOCOL)

# Save the label encoder (if you did class encoding)
with open('label_encoder.pkl', 'wb') as handle:
    pickle.dump(le, handle, protocol=pickle.HIGHEST_PROTOCOL)


from flask import Flask, request, jsonify
import numpy as np
import pickle
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.sequence import pad_sequences

app = Flask(__name__)

# Load artifacts
model = load_model('flipkart_sentiment_model.keras')
with open('tokenizer.pkl', 'rb') as handle:
    tokenizer = pickle.load(handle)
with open('label_encoder.pkl', 'rb') as handle:
    le = pickle.load(handle)

MAX_LEN = 100  # Same as during training

@app.route('/predict', methods=['POST'])
def predict():
    try:
        # Get review text from POST request
        data = request.get_json()
        text = data['text']

        # Preprocess
        seq = tokenizer.texts_to_sequences([text])
        padded = pad_sequences(seq, maxlen=MAX_LEN)

        # Predict
        pred = model.predict(padded)
        sentiment = le.inverse_transform([np.argmax(pred)])[0]
        confidence = float(np.max(pred))

        return jsonify({
            'sentiment': sentiment,
            'confidence': confidence,
            'text': text
        })
    except Exception as e:
        return jsonify({'error': str(e)})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-18-c08a5d55db82> in <cell line: 0>()
      8
      9 # Load artifacts
---> 10 model = load_model('flipkart_sentiment_model.keras')
     11 with open('tokenizer.pkl', 'rb') as handle:
     12     tokenizer = pickle.load(handle)

/usr/local/lib/python3.11/dist-packages/keras/src/saving/saving_api.py in load_model(filepath, custom_objects,
compile, safe_mode)
    198         )
    199     elif str(filepath).endswith(".keras"):
--> 200         raise ValueError(
    201             f"File not found: filepath={filepath}. "
    202             "Please ensure the file is an accessible `.keras` "

ValueError: File not found: filepath=flipkart_sentiment_model.keras. Please ensure the file is an accessible `.keras`
zip file.
```

```
from flask import Flask, request, jsonify
from flask_ngrok import run_with_ngrok
import pickle
import numpy as np
from tensorflow.keras.models import load_model

app = Flask(__name__)
run_with_ngrok(app)  # Start ngrok when app is run

# Load model
model = load_model('/content/flipkart_sentiment_model.keras')

@app.route('/predict', methods=['POST'])
def predict():
    try:
        # Get review text from POST request
        data = request.get_json()
        text = data['text']

        # Preprocess
        seq = tokenizer.texts_to_sequences([text])
        padded = pad_sequences(seq, maxlen=MAX_LEN)

        # Predict
        pred = model.predict(padded)
        sentiment = le.inverse_transform([np.argmax(pred)])[0]
        confidence = float(np.max(pred))

        return jsonify({
            'sentiment': sentiment,
            'confidence': confidence,
            'text': text
        })
    except Exception as e:
        return jsonify({'error': str(e)})

app.run()
```

```
 * Serving Flask app '__main__'
 * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
Exception in thread Thread-10:
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/urllib3/connection.py", line 198, in _new_conn
    sock = connection.create_connection(
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/urllib3/util/connection.py", line 85, in create_connection
    raise err
  File "/usr/local/lib/python3.11/dist-packages/urllib3/util/connection.py", line 73, in create_connection
    sock.connect(sa)
ConnectionRefusedError: [Errno 111] Connection refused

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

```
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/urllib3/connectionpool.py", line 787, in urlopen
    response = self._make_request(
               ^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/urllib3/connectionpool.py", line 493, in _make_request
    conn.request(
  File "/usr/local/lib/python3.11/dist-packages/urllib3/connection.py", line 445, in request
    self.endheaders()
  File "/usr/lib/python3.11/http/client.py", line 1298, in endheaders
    self._send_output(message_body, encode_chunked=encode_chunked)
  File "/usr/lib/python3.11/http/client.py", line 1058, in _send_output
    self.send(msg)
  File "/usr/lib/python3.11/http/client.py", line 996, in send
    self.connect()
  File "/usr/local/lib/python3.11/dist-packages/urllib3/connection.py", line 276, in connect
    self.sock = self._new_conn()
                ^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/urllib3/connection.py", line 213, in _new_conn
    raise NewConnectionError(
urllib3.exceptions.NewConnectionError: <urllib3.connection.HTTPConnection object at 0x7940a726aa90>: Failed to estab

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

Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/requests/adapters.py", line 667, in send
    resp = conn.urlopen(
           ^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/urllib3/connectionpool.py", line 841, in urlopen
    retries = retries.increment(
              ^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/urllib3/util/retry.py", line 519, in increment
    raise MaxRetryError(_pool, url, reason) from reason  # type: ignore[arg-type]
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
urllib3.exceptions.MaxRetryError: HTTPConnectionPool(host='localhost', port=4040): Max retries exceeded with url: /a

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
```

```
from flask import Flask, request, jsonify
from flask_ngrok import run_with_ngrok
import numpy as np
import pickle
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Load necessary pre-trained components
with open('/content/tokenizer.pkl', 'rb') as handle:
    tokenizer = pickle.load(handle)

with open('/content/label_encoder.pkl', 'rb') as handle:
    le = pickle.load(handle)

MAX_LEN = 100  # Set according to your training setup

# Load model
model = load_model('/content/flipkart_sentiment_model.keras')

# Create Flask app
app = Flask(__name__)
run_with_ngrok(app)  # Start ngrok tunnel when the app is run

@app.route('/predict', methods=['POST'])
def predict():
    try:
        # Get input from POST request
        data = request.get_json()
        text = data['text']

        # Preprocess input
        seq = tokenizer.texts_to_sequences([text])
        padded = pad_sequences(seq, maxlen=MAX_LEN)

        # Predict
        pred = model.predict(padded)
```

```
            sentiment = le.inverse_transform([np.argmax(pred)])[0]
            confidence = float(np.max(pred))

            return jsonify({
                'sentiment': sentiment,
                'confidence': confidence,
                'text': text
            })

        except Exception as e:
            return jsonify({'error': str(e)})

    # Run Flask app with ngrok tunnel
    app.run()
```

```
 * Serving Flask app '__main__'
 * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
Exception in thread Thread-12:
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/urllib3/connection.py", line 198, in _new_conn
    sock = connection.create_connection(
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/urllib3/util/connection.py", line 85, in create_connection
    raise err
  File "/usr/local/lib/python3.11/dist-packages/urllib3/util/connection.py", line 73, in create_connection
    sock.connect(sa)
ConnectionRefusedError: [Errno 111] Connection refused

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

Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/urllib3/connectionpool.py", line 787, in urlopen
    response = self._make_request(
               ^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/urllib3/connectionpool.py", line 493, in _make_request
    conn.request(
  File "/usr/local/lib/python3.11/dist-packages/urllib3/connection.py", line 445, in request
    self.endheaders()
  File "/usr/lib/python3.11/http/client.py", line 1298, in endheaders
    self._send_output(message_body, encode_chunked=encode_chunked)
  File "/usr/lib/python3.11/http/client.py", line 1058, in _send_output
    self.send(msg)
  File "/usr/lib/python3.11/http/client.py", line 996, in send
    self.connect()
  File "/usr/local/lib/python3.11/dist-packages/urllib3/connection.py", line 276, in connect
    self.sock = self._new_conn()
                ^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/urllib3/connection.py", line 213, in _new_conn
    raise NewConnectionError(
urllib3.exceptions.NewConnectionError: <urllib3.connection.HTTPConnection object at 0x7940a7e5d110>: Failed to estab

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

Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/requests/adapters.py", line 667, in send
    resp = conn.urlopen(
           ^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/urllib3/connectionpool.py", line 841, in urlopen
    retries = retries.increment(
              ^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/urllib3/util/retry.py", line 519, in increment
    raise MaxRetryError(_pool, url, reason) from reason  # type: ignore[arg-type]
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
urllib3.exceptions.MaxRetryError: HTTPConnectionPool(host='localhost', port=4040): Max retries exceeded with url: /a

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/lib/python3.11/threading.py", line 1045, in _bootstrap_inner
```

Start coding or generate with AI.

```
!pip install flask-ngrok # Install the flask_ngrok package
```

```
Collecting flask-ngrok
  Downloading flask_ngrok-0.0.25-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: Flask>=0.8 in /usr/local/lib/python3.11/dist-packages (from flask-ngrok) (3.1.0)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from flask-ngrok) (2.32.3)
Requirement already satisfied: Werkzeug>=3.1 in /usr/local/lib/python3.11/dist-packages (from Flask>=0.8->flask-ngrok)
Requirement already satisfied: Jinja2>=3.1.2 in /usr/local/lib/python3.11/dist-packages (from Flask>=0.8->flask-ngrok)
Requirement already satisfied: itsdangerous>=2.2 in /usr/local/lib/python3.11/dist-packages (from Flask>=0.8->flask-ng
Requirement already satisfied: click>=8.1.3 in /usr/local/lib/python3.11/dist-packages (from Flask>=0.8->flask-ngrok)
Requirement already satisfied: blinker>=1.9 in /usr/local/lib/python3.11/dist-packages (from Flask>=0.8->flask-ngrok)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->fla
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->flask-ngrok) (3
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->flask-ngr
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->flask-ngr
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from Jinja2>=3.1.2->Flask>=
Downloading flask_ngrok-0.0.25-py3-none-any.whl (3.1 kB)
Installing collected packages: flask-ngrok
Successfully installed flask-ngrok-0.0.25
```

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Conv1D, GlobalMaxPooling1D, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from kerastuner import HyperModel, RandomSearch

# 1. Enhanced HyperModel Definition
class OptimizedSentimentHyperModel(HyperModel):
    def build(self, hp):
        model = Sequential([
            # Smaller embedding dimension
            Embedding(input_dim=vocab_size,
                      output_dim=hp.Int('embedding_dim', 64, 128, step=32),
                      input_length=max_len),

            # Fewer filters with regularization
            Conv1D(
                filters=hp.Int('conv_filters', 64, 128, step=32),
                kernel_size=hp.Choice('kernel_size', [3, 5]),
                activation='relu',
                padding='same',
                kernel_regularizer='l2'  # Added L2 regularization
            ),

            GlobalMaxPooling1D(),

            # Smaller dense layer with higher dropout
            Dense(
                units=hp.Int('dense_units', 32, 64, step=16),
                activation='relu',
                kernel_regularizer='l2'
            ),

            # Increased dropout rate
            Dropout(rate=hp.Float('dropout_rate', 0.5, 0.7, step=0.1)),

            Dense(3, activation='softmax')
        ])

        optimizer = hp.Choice('optimizer', ['adam', 'rmsprop'])
        model.compile(
            optimizer=optimizer,
            loss='sparse_categorical_crossentropy',
            metrics=['accuracy']
        )
        return model

# 2. Tuner Setup with Early Stopping
tuner = RandomSearch(
    OptimizedSentimentHyperModel(),
    objective='val_accuracy',
    max_trials=15,
    executions_per_trial=2,
```

```
    directory='optimized_tuning',
    project_name='sentiment_analysis_v2'
)

early_stop = EarlyStopping(
    monitor='val_loss',
    patience=5,
    restore_best_weights=True  # Critical for getting best model
)

# 3. Run Training with Validation Split
tuner.search(
    X_train_pad, y_train,
    epochs=50,  # Higher but will stop early
    batch_size=32,
    validation_split=0.3,  # Increased validation size
    callbacks=[early_stop],
    verbose=2
)

# 4. Retrieve and Evaluate Best Model
best_model = tuner.get_best_models(num_models=1)[0]
best_hps = tuner.get_best_hyperparameters()[0]

# Evaluate
test_loss, test_acc = best_model.evaluate(X_test_pad, y_test, verbose=0)
print(f"\nOptimized Model Test Accuracy: {test_acc:.4f}")
print(f"Test Loss: {test_loss:.4f}")

# 5. Save the Optimized Model
best_model.save('optimized_sentiment_model.keras')

# Classification Report
y_pred = np.argmax(best_model.predict(X_test_pad), axis=1)
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=le.classes_))
```

```
Trial 15 Complete [00h 00m 57s]
val_accuracy: 0.8111588060855865

Best val_accuracy So Far: 0.8326180279254913
Total elapsed time: 00h 11m 44s
/usr/local/lib/python3.11/dist-packages/keras/src/saving/saving_lib.py:757: UserWarning: Skipping variable loading for
  saveable.load_own_variables(weights_store.get(inner_path))

Optimized Model Test Accuracy: 0.7371
Test Loss: 0.7584
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step

Classification Report:
              precision    recall  f1-score   support

    negative       0.83      0.81      0.82       123
     neutral       0.52      0.79      0.63        29
    positive       0.67      0.48      0.56        42

    accuracy                           0.74       194
   macro avg       0.67      0.69      0.67       194
weighted avg       0.75      0.74      0.74       194
```

```
!pip install keras-tuner # Install the keras-tuner package
```

```
Collecting keras-tuner
  Downloading keras_tuner-1.4.7-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: keras in /usr/local/lib/python3.11/dist-packages (from keras-tuner) (3.8.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from keras-tuner) (24.2)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from keras-tuner) (2.32.3)
Collecting kt-legacy (from keras-tuner)
  Downloading kt_legacy-1.0.5-py3-none-any.whl.metadata (221 bytes)
Requirement already satisfied: absl-py in /usr/local/lib/python3.11/dist-packages (from keras->keras-tuner) (1.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from keras->keras-tuner) (2.0.2)
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from keras->keras-tuner) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (from keras->keras-tuner) (0.0.8)
Requirement already satisfied: h5py in /usr/local/lib/python3.11/dist-packages (from keras->keras-tuner) (3.13.0)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (from keras->keras-tuner) (0.15.0)
```

```
Requirement already satisfied: ml-dtypes in /usr/local/lib/python3.11/dist-packages (from keras->keras-tuner) (0.4.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->ker
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->keras-tuner) (3
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->keras-tun
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->keras-tun
Requirement already satisfied: typing-extensions>=4.5.0 in /usr/local/lib/python3.11/dist-packages (from optree->keras
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras->ker
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras->k
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich
Downloading keras_tuner-1.4.7-py3-none-any.whl (129 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 129.1/129.1 kB 2.4 MB/s eta 0:00:00
Downloading kt_legacy-1.0.5-py3-none-any.whl (9.6 kB)
Installing collected packages: kt-legacy, keras-tuner
Successfully installed keras-tuner-1.4.7 kt-legacy-1.0.5
```

```
!pip install --upgrade scikit-learn scikeras
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: scikeras in /usr/local/lib/python3.11/dist-packages (0.13.0)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.
Requirement already satisfied: keras>=3.2.0 in /usr/local/lib/python3.11/dist-packages (from scikeras) (3.8.0)
Requirement already satisfied: absl-py in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (1.4.0
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (0.0.8)
Requirement already satisfied: h5py in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (3.13.0)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (0.14.1
Requirement already satisfied: ml-dtypes in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (0.4
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (24.
Requirement already satisfied: typing-extensions>=4.5.0 in /usr/local/lib/python3.11/dist-packages (from optree->keras
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.2
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich
```

```
import scikeras
import sklearn
import tensorflow as tf

print("SciKeras version:", scikeras.__version__)
print("Scikit-learn version:", sklearn.__version__)
print("TensorFlow version:", tf.__version__)
```

```
SciKeras version: 0.13.0
Scikit-learn version: 1.6.1
TensorFlow version: 2.18.0
```

Start coding or generate with AI.

```
!pip install --upgrade scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.
```

```
pip install scikeras
```

```
Collecting scikeras
  Downloading scikeras-0.13.0-py3-none-any.whl.metadata (3.1 kB)
Requirement already satisfied: keras>=3.2.0 in /usr/local/lib/python3.11/dist-packages (from scikeras) (3.8.0)
Requirement already satisfied: scikit-learn>=1.4.2 in /usr/local/lib/python3.11/dist-packages (from scikeras) (1.6.1)
Requirement already satisfied: absl-py in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (1.4.0
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (2.0.2)
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (0.0.8)
Requirement already satisfied: h5py in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (3.13.0)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (0.14.1
Requirement already satisfied: ml-dtypes in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (0.4
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (24.
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.4.2->scik
```

```
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.4.2->sci
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.4
Requirement already satisfied: typing-extensions>=4.5.0 in /usr/local/lib/python3.11/dist-packages (from optree->keras
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.2
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich
Downloading scikeras-0.13.0-py3-none-any.whl (26 kB)
Installing collected packages: scikeras
Successfully installed scikeras-0.13.0
```

```
#SBERT FLLOWED BY XGBOOST
from sentence_transformers import SentenceTransformer
from xgboost import XGBClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report
import pandas as pd

# Example data (replace with your actual train/test sets)
# train and test should have 'text' and 'sentiment_label' columns
# train = pd.read_csv(...)
# test = pd.read_csv(...)

# 1. Encode sentiment labels
le = LabelEncoder()
train_labels = le.fit_transform(train['sentiment_label'])
test_labels = le.transform(test['sentiment_label'])

# 2. Use SBERT to generate embeddings
sbert_model = SentenceTransformer('all-MiniLM-L6-v2')
X_train = sbert_model.encode(train['Content'].tolist(), show_progress_bar=True)
X_test = sbert_model.encode(test['Content'].tolist(), show_progress_bar=True)

# 3. Train XGBoost classifier
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')
xgb_model.fit(X_train, train_labels)

# 4. Predict and evaluate
y_pred = xgb_model.predict(X_test)
y_pred_labels = le.inverse_transform(y_pred)
true_labels = le.inverse_transform(test_labels)

# 5. Classification report
print(classification_report(true_labels, y_pred_labels))
```

```
Batches: 100%                                                    25/25 [00:27<00:00,  4.18it/s]
Batches: 100%                                                    7/7 [00:08<00:00,  1.35it/s]
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [17:22:58] WARNING: /workspace/src/learner.c
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
              precision    recall  f1-score   support

    negative       0.74      0.93      0.82       123
     neutral       0.76      0.45      0.57        29
    positive       0.57      0.29      0.38        42

    accuracy                           0.72       194
   macro avg       0.69      0.56      0.59       194
weighted avg       0.71      0.72      0.69       194
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, GRU
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Tokenization
tokenizer = Tokenizer(num_words=10000, oov_token='<OOV>')
tokenizer.fit_on_texts(train['Content'])
X_train_seq = tokenizer.texts_to_sequences(train['Content'])
X_test_seq = tokenizer.texts_to_sequences(test['Content'])

# Padding
```

```
max_len = 100
X_train_pad = pad_sequences(X_train_seq, maxlen=max_len, padding='post')
X_test_pad = pad_sequences(X_test_seq, maxlen=max_len, padding='post')

# Encode labels
le = LabelEncoder()
y_train = le.fit_transform(train['sentiment_label'])
y_test = le.transform(test['sentiment_label'])

# Model
model = Sequential([
    Embedding(input_dim=10000, output_dim=128, input_length=max_len),
    LSTM(64, return_sequences=False),  # or GRU(64)
    Dense(3, activation='softmax')
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train_pad, y_train, validation_split=0.1, epochs=5)

# Evaluate
y_pred = model.predict(X_test_pad).argmax(axis=1)
print(classification_report(y_test, y_pred))
```

```
Epoch 1/5
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is
  warnings.warn(
22/22 ━━━━━━━━━━━━━━━━━━━━ 10s 166ms/step - accuracy: 0.5529 - loss: 1.0057 - val_accuracy: 0.6154 - val_loss: 0.9400
Epoch 2/5
22/22 ━━━━━━━━━━━━━━━━━━━━ 3s 78ms/step - accuracy: 0.6257 - loss: 0.9323 - val_accuracy: 0.6154 - val_loss: 0.9335
Epoch 3/5
22/22 ━━━━━━━━━━━━━━━━━━━━ 2s 79ms/step - accuracy: 0.6053 - loss: 0.9531 - val_accuracy: 0.6154 - val_loss: 0.9294
Epoch 4/5
22/22 ━━━━━━━━━━━━━━━━━━━━ 3s 124ms/step - accuracy: 0.6041 - loss: 0.9562 - val_accuracy: 0.6154 - val_loss: 0.9258
Epoch 5/5
22/22 ━━━━━━━━━━━━━━━━━━━━ 2s 76ms/step - accuracy: 0.6343 - loss: 0.9192 - val_accuracy: 0.6154 - val_loss: 0.9292
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
              precision    recall  f1-score   support

           0       0.63      1.00      0.78       123
           1       0.00      0.00      0.00        29
           2       0.00      0.00      0.00        42

    accuracy                           0.63       194
   macro avg       0.21      0.33      0.26       194
weighted avg       0.40      0.63      0.49       194

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
!pip install fasttext
```

```
Collecting fasttext
  Downloading fasttext-0.9.3.tar.gz (73 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 73.4/73.4 kB 1.9 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting pybind11>=2.2 (from fasttext)
  Using cached pybind11-2.13.6-py3-none-any.whl.metadata (9.5 kB)
Requirement already satisfied: setuptools>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from fasttext) (75.2.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from fasttext) (2.0.2)
Using cached pybind11-2.13.6-py3-none-any.whl (243 kB)
Building wheels for collected packages: fasttext
  Building wheel for fasttext (pyproject.toml) ... done
  Created wheel for fasttext: filename=fasttext-0.9.3-cp311-cp311-linux_x86_64.whl size=4313502 sha256=78d5e8d055eaa19
  Stored in directory: /root/.cache/pip/wheels/65/4f/35/5057db0249224e9ab55a513fa6b79451473ceb7713017823c3
Successfully built fasttext
Installing collected packages: pybind11, fasttext
Successfully installed fasttext-0.9.3 pybind11-2.13.6
```

```
!pip install --upgrade pymongo certifi
```

```
Requirement already satisfied: pymongo in /usr/local/lib/python3.11/dist-packages (4.11.3)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (2025.1.31)
Requirement already satisfied: dnspython<3.0.0,>=1.16.0 in /usr/local/lib/python3.11/dist-packages (from pymongo) (2.7
```

```
import sys

print(sys.version)
```

```
3.11.11 (main, Dec  4 2024, 08:55:07) [GCC 11.4.0]
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

# 1. Prepare Data ------------------------------------------------
# Convert to FastText format (__label__prefix required)
df_cleaned['fasttext_format'] = df_cleaned.apply(
    lambda row: f"__label__{row['sentiment_label']} {row['Content']}",
    axis=1
)

# Split data (80% train, 20% test)
train, test = train_test_split(df_cleaned, test_size=0.2, random_state=42)
# Sample: Assume you have 'review_text' and 'rating' columns
df = pd.read_csv('trustpilotFlipkartReviews.csv')

# Clean data (optional: remove NAs, filter non-English, etc.)
df.dropna(subset=['Content', 'Rating'], inplace=True)

# Convert to string and categorical class
X = df['Content'].astype(str)
y = df['Rating'].astype(int)  # Ratings should be 1-5

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)

# TF-IDF Vectorisation
vectorizer = TfidfVectorizer(max_features=5000, ngram_range=(1,2))
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

# Model: Logistic Regression
model = LogisticRegression(multi_class='multinomial', solver='saga', max_iter=1000)
model.fit(X_train_tfidf, y_train)

# Predict & Evaluate
y_pred = model.predict(X_test_tfidf)
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
<ipython-input-9-a86e562ed0f8>:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-
  df_cleaned['fasttext_format'] = df_cleaned.apply(
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning: 'multi_class' was depre
  warnings.warn(
              precision    recall  f1-score   support

           1       0.89      1.00      0.94       172
           2       0.00      0.00      0.00         4
           3       0.00      0.00      0.00         3
           4       0.00      0.00      0.00         5
           5       0.00      0.00      0.00        10

    accuracy                           0.89       194
   macro avg       0.18      0.20      0.19       194
weighted avg       0.79      0.89      0.83       194
```

```
[[172   0   0   0   0]
 [  4   0   0   0   0]
 [  3   0   0   0   0]
 [  5   0   0   0   0]
 [ 10   0   0   0   0]]
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
from xgboost import XGBClassifier
from sklearn.metrics import classification_report
from imblearn.over_sampling import SMOTE

# Fix class labels to start from 0
y_train_encoded = y_train - 1
y_test_encoded = y_test - 1

# Oversample minority classes
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train_tfidf, y_train_encoded)

# XGBoost model
model = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')
model.fit(X_train_resampled, y_train_resampled)

# Evaluate
y_pred = model.predict(X_test_tfidf)
print(classification_report(y_test, y_pred+1))
```

```
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [23:14:09] WARNING: /workspace/src/learner.c
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
              precision    recall  f1-score   support

           1       0.91      0.99      0.95       172
           2       0.00      0.00      0.00         4
           3       1.00      0.33      0.50         3
           4       0.00      0.00      0.00         5
           5       0.50      0.30      0.38        10

    accuracy                           0.90       194
   macro avg       0.48      0.32      0.36       194
weighted avg       0.85      0.90      0.87       194

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from imblearn.over_sampling import SMOTE
from collections import Counter

# 1. Load and clean data ----------------------------------------------------
df = pd.read_csv('trustpilotFlipkartReviews.csv')
df.dropna(subset=['Content', 'Rating'], inplace=True)
df['Content'] = df['Content'].astype(str)
df['Rating'] = df['Rating'].astype(int)

# Optional: check class distribution
print("Class distribution before resampling:", Counter(df['Rating']))
```

```
# 2. Split the data with stratification -----------------------------------
X = df['Content']
y = df['Rating']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

# 3. TF-IDF Vectorisation -------------------------------------------------
vectorizer = TfidfVectorizer(max_features=5000, ngram_range=(1,2))
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

# 4. Address Class Imbalance with SMOTE -----------------------------------
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train_tfidf, y_train)

print("Class distribution after SMOTE:", Counter(y_train_resampled))

# 5. Train the model with balanced weights --------------------------------
model = LogisticRegression(
    multi_class='multinomial',
    solver='saga',
    max_iter=1000,
    class_weight='balanced'
)
model.fit(X_train_resampled, y_train_resampled)

# 6. Predict and Evaluate -------------------------------------------------
y_pred = model.predict(X_test_tfidf)

print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Class distribution before resampling: Counter({1: 860, 5: 48, 4: 25, 2: 20, 3: 17})
Class distribution after SMOTE: Counter({1: 688, 3: 688, 4: 688, 5: 688, 2: 688})
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning: 'multi_class' was depre
  warnings.warn(

Classification Report:
              precision    recall  f1-score   support

           1       0.91      0.99      0.95       172
           2       0.00      0.00      0.00         4
           3       0.00      0.00      0.00         3
           4       0.00      0.00      0.00         5
           5       0.60      0.30      0.40        10

    accuracy                           0.89       194
   macro avg       0.30      0.26      0.27       194
weighted avg       0.84      0.89      0.86       194


Confusion Matrix:
 [[170   0   0   0   2]
 [  4   0   0   0   0]
 [  3   0   0   0   0]
 [  5   0   0   0   0]
 [  5   0   2   0   3]]
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Conv1D, GlobalMaxPooling1D, Dense, Dropout
```

```
from tensorflow.keras.utils import to_categorical

# Load and prepare data
df = pd.read_csv('trustpilotFlipkartReviews.csv')
df.dropna(subset=['Content', 'Rating'], inplace=True)
df['Content'] = df['Content'].astype(str)
df['Rating'] = df['Rating'].astype(int)

# Encode labels starting from 0
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(df['Rating'])  # Now 0 to 4 instead of 1 to 5
num_classes = len(np.unique(y))
y_cat = to_categorical(y, num_classes=num_classes)

# Tokenise text
tokenizer = Tokenizer(num_words=10000)
tokenizer.fit_on_texts(df['Content'])
X_seq = tokenizer.texts_to_sequences(df['Content'])

# Pad sequences
max_len = 200
X_pad = pad_sequences(X_seq, maxlen=max_len)

# Split data
X_train, X_test, y_train, y_test = train_test_split(X_pad, y_cat, test_size=0.2, stratify=y, random_state=42)

# Build CNN model
model = Sequential()
model.add(Embedding(input_dim=10000, output_dim=128, input_length=max_len))
model.add(Conv1D(128, kernel_size=5, activation='relu'))
model.add(GlobalMaxPooling1D())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

# Compile model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train model
model.fit(X_train, y_train, epochs=5, batch_size=32, validation_split=0.1)

# Evaluate
loss, accuracy = model.evaluate(X_test, y_test)
print(f"\nTest Accuracy: {accuracy:.4f}")
# Predict class probabilities
y_pred_probs = model.predict(X_test)

# Convert probabilities to class labels
y_pred_classes = np.argmax(y_pred_probs, axis=1)
y_true_classes = np.argmax(y_test, axis=1)

# Inverse transform the labels to get original ratings (if needed)
y_pred_labels = label_encoder.inverse_transform(y_pred_classes)
y_true_labels = label_encoder.inverse_transform(y_true_classes)

# Print classification report
print("\nClassification Report:\n")
print(classification_report(y_true_labels, y_pred_labels))
```

```
Epoch 1/5
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is
  warnings.warn(
22/22 ━━━━━━━━━━━━━━━━━━━━ 6s 128ms/step - accuracy: 0.6640 - loss: 1.1822 - val_accuracy: 0.8718 - val_loss: 0.7251
Epoch 2/5
22/22 ━━━━━━━━━━━━━━━━━━━━ 3s 114ms/step - accuracy: 0.8986 - loss: 0.5189 - val_accuracy: 0.8718 - val_loss: 0.5421
Epoch 3/5
22/22 ━━━━━━━━━━━━━━━━━━━━ 4s 88ms/step - accuracy: 0.8895 - loss: 0.4652 - val_accuracy: 0.8718 - val_loss: 0.5296
Epoch 4/5
22/22 ━━━━━━━━━━━━━━━━━━━━ 2s 90ms/step - accuracy: 0.8975 - loss: 0.4048 - val_accuracy: 0.8718 - val_loss: 0.5129
Epoch 5/5
22/22 ━━━━━━━━━━━━━━━━━━━━ 3s 92ms/step - accuracy: 0.8911 - loss: 0.3860 - val_accuracy: 0.8718 - val_loss: 0.4922
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - accuracy: 0.8660 - loss: 0.4932

Test Accuracy: 0.8866
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
```

```
Classification Report:

              precision    recall  f1-score   support

           1       0.89      1.00      0.94       172
           2       0.00      0.00      0.00         4
           3       0.00      0.00      0.00         3
           4       0.00      0.00      0.00         5
           5       0.00      0.00      0.00        10

    accuracy                           0.89       194
   macro avg       0.18      0.20      0.19       194
weighted avg       0.79      0.89      0.83       194

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
unique_values_count = df_cleaned['Rating'].nunique()
print("Number of unique values in 'ratings':", unique_values_count)
```

```
Number of unique values in 'ratings': 5
```

```
# Count of rows per unique value in 'ratings'
counts = df_cleaned['Rating'].value_counts()
print(counts)
```

```
Rating
1.0    859
5.0     48
4.0     25
2.0     20
3.0     17
Name: count, dtype: int64
```

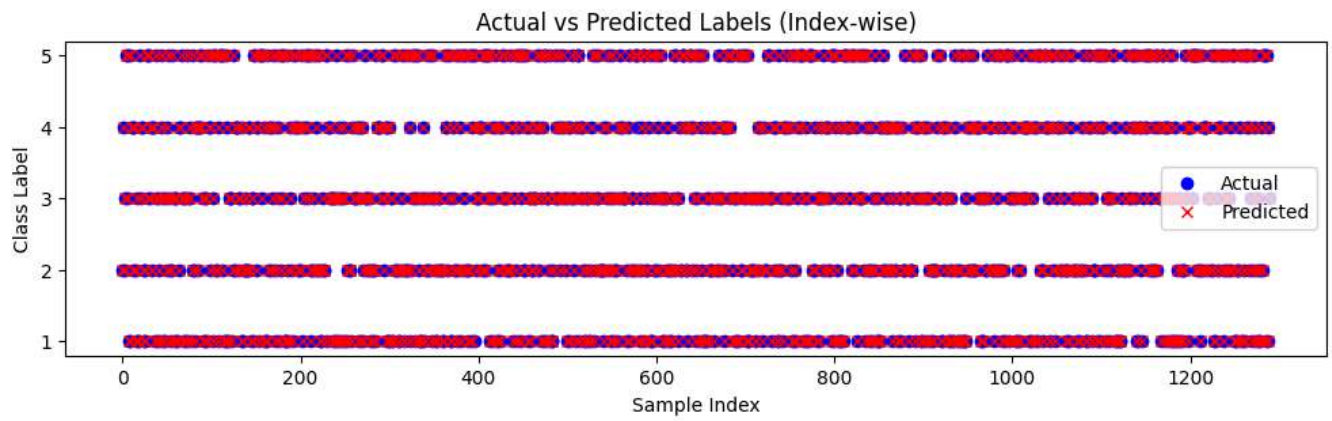
```
from sklearn.feature_extraction.text import TfidfVectorizer
from imblearn.combine import SMOTEENN
from scipy.sparse import hstack
df = pd.read_csv('trustpilotFlipkartReviews.csv')

# Clean data (optional: remove NAs, filter non-English, etc.)
df.dropna(subset=['Content', 'Rating'], inplace=True)

# Convert to string and categorical class
X = df['Content'].astype(str)
y = df['Rating'].astype(int)  # Ratings should be 1–5
# Assuming your text column is named 'review_text'
tfidf = TfidfVectorizer(max_features=5000)  # You can adjust max_features

X_text = tfidf.fit_transform(df['Content'])



# If only text features:
X_combined = X_text

smote = SMOTE(random_state=42)
X_res, y_res = smote.fit_resample(X_combined, y)
print(y_res.value_counts())
```

```
Rating
5    860
1    860
3    860
2    860
4    860
Name: count, dtype: int64
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix

# 1. Split the resampled data
X_train, X_test, y_train, y_test = train_test_split(X_res, y_res, test_size=0.3, random_state=42)

# 2. Train a classifier
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)

# 3. Predict
y_pred = clf.predict(X_test)

# 4. Evaluate
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Classification Report:
                precision    recall  f1-score   support

           1       0.98      0.99      0.98       261
           2       1.00      1.00      1.00       278
           3       1.00      1.00      1.00       250
           4       0.99      1.00      0.99       232
           5       1.00      0.99      0.99       269

    accuracy                           0.99      1290
   macro avg       0.99      0.99      0.99      1290
weighted avg       0.99      0.99      0.99      1290

Confusion Matrix:
 [[258   0   0   2   1]
 [  1 277   0   0   0]
 [  1   0 249   0   0]
 [  1   0   0 231   0]
 [  3   0   0   0 266]]
```

```
import numpy as np

plt.figure(figsize=(12, 3))
plt.plot(np.arange(len(y_test)), y_test, 'bo', label='Actual')
plt.plot(np.arange(len(y_pred)), y_pred, 'rx', label='Predicted')
plt.legend()
plt.title("Actual vs Predicted Labels (Index-wise)")
plt.xlabel("Sample Index")
plt.ylabel("Class Label")
plt.show()
```




```
import seaborn as sns
import pandas as pd

df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})

plt.figure(figsize=(10,5))
sns.countplot(x='Actual', data=df, color='blue', alpha=0.6, label='Actual')
sns.countplot(x='Predicted', data=df, color='red', alpha=0.4, label='Predicted')
```

```
plt.title("Actual vs Predicted Class Distribution")
plt.legend()
plt.show()
```