Gemini

```
# Install required libraries
# !pip install pandas scikit-learn nltk spacy
# !python -m spacy download en_core_web_sm

import pandas as pd
import nltk
import spacy
import string

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# ==========================
# NLTK Setup
# ==========================
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('punkt_tab')


# ==========================
# spaCy Setup
# ==========================
nlp = spacy.load("en_core_web_sm")

# ==========================
# Step 1: Load Dataset
# ==========================
# Download the file manually from Google Drive, then load it:
try:
    data = pd.read_csv("Stock Headlines.csv", encoding='utf-8')  # Try UTF-8 first
except UnicodeDecodeError:
    data = pd.read_csv("Stock Headlines.csv", encoding='latin1') # If UTF-8 fails, try latin1

# Use only first 100 rows
data = data.head(100)

# Check if 'Top1' and 'Label' columns exist
if 'Top1' not in data.columns:
    raise KeyError("Column 'Top1' not found in the DataFrame. Please check your CSV file and column names.")
if 'Label' not in data.columns:
    raise KeyError("Column 'Label' not found in the DataFrame. Please check your CSV file and column names.")

# ==========================
# Step 2: Preprocess Text
# ==========================
def preprocess_text(text):
    text = str(text).lower()  # Lowercase
    text = text.translate(str.maketrans('', '', string.punctuation))  # Remove punctuation

    # NLTK Tokenization
    tokens = word_tokenize(text)

    # Remove stopwords
    tokens = [word for word in tokens if word not in set(stopwords.words('english'))]

    # spaCy Lemmatization
    doc = nlp(" ".join(tokens))
    lemmatized_tokens = [token.lemma_ for token in doc]

    return " ".join(lemmatized_tokens)

# Apply preprocessing on the text column (replace 'headline' with your column name)
data['clean_text'] = data['Top1'].apply(preprocess_text)

# ==========================
# Step 3: Prepare Features and Labels
# ==========================
# Replace 'buy' with your target column
X = data['clean_text']
y = data['Label']  # Assuming 'buy' column is 0 or 1

# Convert text to numerical features
vectorizer = CountVectorizer()
```

```
vectorizer = CountVectorizer()
X_vect = vectorizer.fit_transform(X)

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X_vect, y, test_size=0.2, random_state=42)

# ==========================
# Step 4: Train Logistic Regression
# ==========================
model = LogisticRegression()
model.fit(X_train, y_train)

# ==========================
# Step 5: Make Predictions
# ==========================
y_pred = model.predict(X_test)

# ==========================
# Step 6: Evaluate Model
# ==========================
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
Accuracy: 0.5
Confusion Matrix:
 [[8 2]
 [8 2]]
Classification Report:
              precision    recall  f1-score   support

           0       0.50      0.80      0.62        10
           1       0.50      0.20      0.29        10

    accuracy                           0.50        20
   macro avg       0.50      0.50      0.45        20
weighted avg       0.50      0.50      0.45        20
```