Data Labeling Python

1.Installing Python PIP

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

```
Requirement already satisfied: transformers in /usr/local/lib/python3.12/dist-packages (4.55.2)
Requirement already satisfied: torch in /usr/local/lib/python3.12/dist-packages (2.8.0+cu126)
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (2.2.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from transformers) (3.19.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.34.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.34.4)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.12/dist-packages (from transformers) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (25.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.12/dist-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from transformers) (2.32.4)
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.21.4)
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.6.2)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.12/dist-packages (from transformers) (4.67.1)
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.12/dist-packages (from torch) (4.14.1)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from torch) (75.2.0)
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch) (1.13.3)
Requirement already satisfied: networkx in /usr/local/lib/python3.12/dist-packages (from torch) (3.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from torch) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.12/dist-packages (from torch) (2025.3.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.80)
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in /usr/local/lib/python3.12/dist-packages (from torch) (9.10.2.21)
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.4.1)
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in /usr/local/lib/python3.12/dist-packages (from torch) (11.3.0.4)
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in /usr/local/lib/python3.12/dist-packages (from torch) (10.3.7.77)
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in /usr/local/lib/python3.12/dist-packages (from torch) (11.7.1.2)
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in /usr/local/lib/python3.12/dist-packages (from torch) (12.5.4.2)
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in /usr/local/lib/python3.12/dist-packages (from torch) (0.7.1)
Requirement already satisfied: nvidia-nccl-cu12==2.27.3 in /usr/local/lib/python3.12/dist-packages (from torch) (2.27.3)
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.85)
```

2.Importing the data set

```
from google.colab import files
import pandas as pd

# Upload the file
uploaded = files.upload()

# Use the correct file name as a string
file_name = "youtubeSpamCollection.csv"  # <-- Put quotes around the file name

try:
    # Read the CSV file into a DataFrame
    df = pd.read_csv(file_name)
    print("File loaded successfully!")
    print(df.head())
except FileNotFoundError:
    print(f"Error: The file '{file_name}' was not found.")
except pd.errors.EmptyDataError:
    print(f"Error: The file '{file_name}' is empty.")
except pd.errors.ParserError:
    print(f"Error: The file '{file_name}' could not be parsed as CSV.")
```

```
Choose Files  No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving youtubeSpamCollection.csv to youtubeSpamCollection (1).csv
File loaded successfully!
output actions
                                  COMMENT_ID            AUTHOR  \
0  LZQPQhLyRh80UYxNuaDWhIGQYNQ96IuCg-AYWqNPjpU        Julius NM
1  LZQPQhLyRh_C2cTtd9MvFRJedxydaVW-2sNg5Diuo4A       adam riyati
2  LZQPQhLyRh9MSZYnf8djyk0gEF9BHDPYrrK-qCczIY8  Evgeny Murashkin
3         z13jhp0bxqncu512g22wvzkasxmvvzjaz04   ElNino Melendez
4         z13fwbwp1oujthgqj04chlngpvzmtt3r3dw           GsMega

                  DATE                                            CONTENT  \
0  2013-11-07T06:20:48  Huh, anyway check out this you[tube] channel: ...
1  2013-11-07T12:37:15  Hey guys check out my new channel and our firs...
2  2013-11-08T17:34:21            just for test I have to say murdev.com
3  2013-11-09T08:28:43  me shaking my sexy ass on my channel enjoy ^_^...
4  2013-11-10T16:05:38         watch?v=vtaRGgvGtWQ   Check this out .Ï»¿

   CLASS
0      1
1      1
2      1
3      1
4      1
```

3.Preporcessing & Labelling The Data

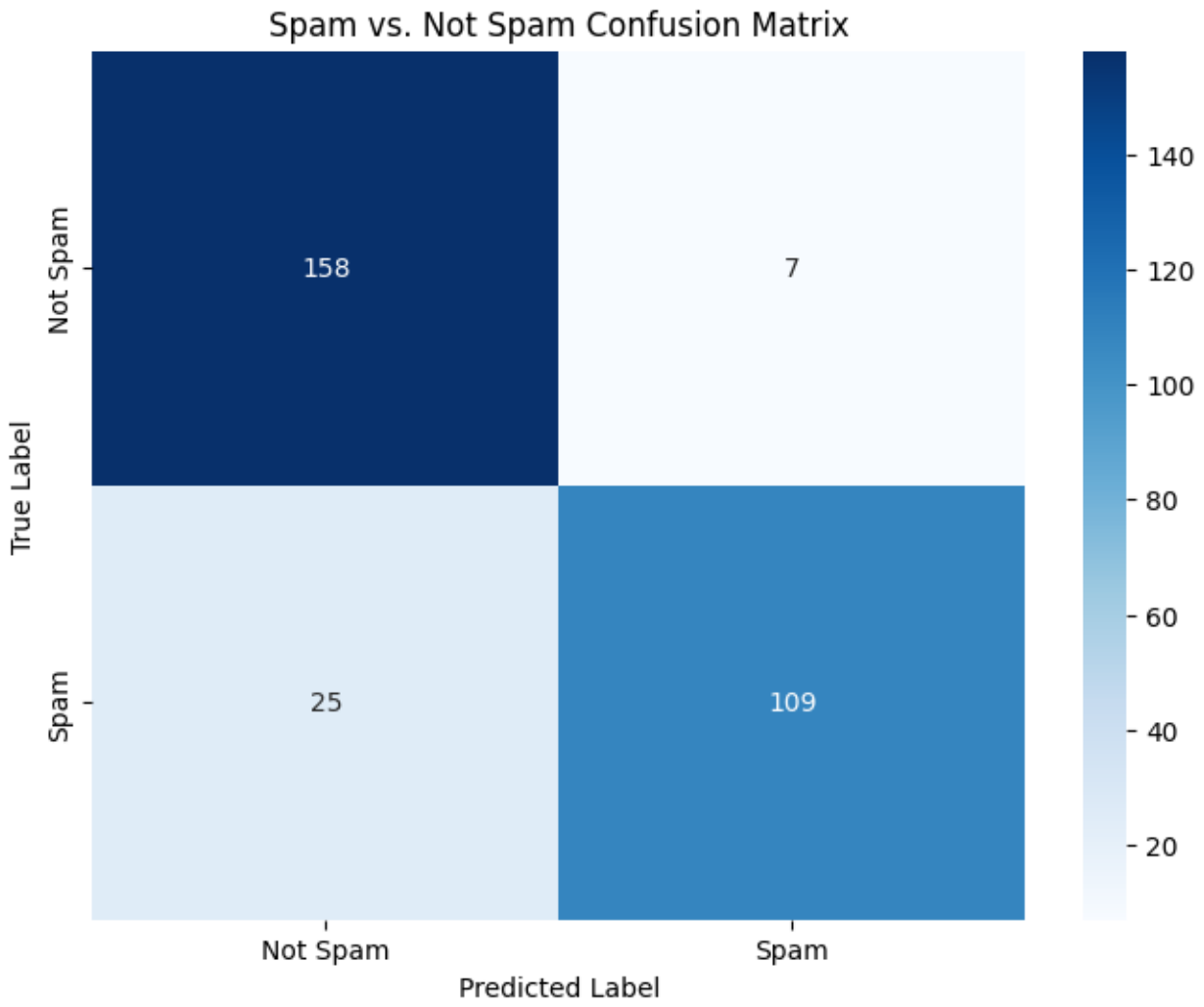
```python
#Preprocessing & Labeling
import re
import string
import nltk
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Download NLTK stopwords
nltk.download('stopwords')

# The 'youtubeSpamCollection.csv' dataset typically has 'CONTENT' and 'CLASS' columns.
# 'CLASS' is the label: 1 for spam, 0 for not spam.

# --- Data Cleaning and Preprocessing ---
stop_words = set(stopwords.words('english'))
```

```python
def clean_text(text):
    text = text.lower()  # Lowercase the text
    text = re.sub(r'https?://\S+|www\.\S+', '', text)  # Remove URLs
    text = re.sub(f'[{re.escape(string.punctuation)}]', '', text)  # Remove punctuation
    text = ' '.join([word for word in text.split() if word not in stop_words])  # Remove stopwords
    return text

# Apply the cleaning function
df['cleaned_content'] = df['CONTENT'].apply(clean_text)
print("\nDataFrame after cleaning:")
print(df[['CONTENT', 'cleaned_content']].head())

# --- Model Training and Labeling ---
X = df['cleaned_content']
y = df['CLASS']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a machine learning pipeline
# TfidfVectorizer converts text into a matrix of TF-IDF features
# MultinomialNB is a good classifier for text data
model = make_pipeline(TfidfVectorizer(), MultinomialNB())

# Train the model
model.fit(X_train, y_train)

# Use the trained model to predict labels on the test set
y_pred = model.predict(X_test)
```

```python
# --- Final Analysis and Visualization ---
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=['Not Spam', 'Spam']))

# Generate and plot a confusion matrix to visualize performance
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Not Spam', 'Spam'], yticklabels=['Not Spam', 'Spam'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Spam vs. Not Spam Confusion Matrix')
plt.show()

# Add the predicted labels to the original DataFrame
df['predicted_label'] = model.predict(df['cleaned_content'])
print("\nFinal DataFrame with Predicted Labels:")
print(df[['CONTENT', 'CLASS', 'predicted_label']].head())
```

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

DataFrame after cleaning:
                                             CONTENT  \
0  Huh, anyway check out this you[tube] channel: ...
1  Hey guys check out my new channel and our firs...
2            just for test I have to say murdev.com
3  me shaking my sexy ass on my channel enjoy ^_^...
4         watch?v=vtaRGgvGtWQ   Check this out .ï»¿

                                cleaned_content
0       huh anyway check youtube channel kobyoshi02
1  hey guys check new channel first vid us monkey...
2                             test say murdevcom
```

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

    Not Spam       0.86      0.96      0.91       165
        Spam       0.94      0.81      0.87       134

    accuracy                           0.89       299
   macro avg       0.90      0.89      0.89       299
weighted avg       0.90      0.89      0.89       299
```



Final DataFrame with Predicted Labels:

	CONTENT	CLASS	predicted_label
0	Huh, anyway check out this you[tube] channel: ...	1	1
1	Hey guys check out my new channel and our firs...	1	1
2	just for test I have to say murdev.com	1	1

```python
import re
import string
import nltk
from nltk.corpus import stopwords
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
import warnings
warnings.filterwarnings('ignore')

# Ensure stopwords are downloaded
nltk.download('stopwords')

# --- Check if DataFrame exists and has necessary columns ---
try:
    X = df['cleaned_content']
    y = df['CLASS']
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42
    )
    print("Data split complete.")
except NameError:
    raise NameError("DataFrame 'df' not found. Please load and clean your data first.")
except KeyError:
    raise KeyError("Required columns 'cleaned_content' or 'CLASS' not found in DataFrame.")
```

```python
# --- Logistic Regression Model ---
print("\n--- Training and evaluating Logistic Regression ---")
lr_pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words=stopwords.words('english'))),
    ('lr', LogisticRegression(random_state=42, max_iter=1000))
])
lr_pipeline.fit(X_train, y_train)
y_pred_lr = lr_pipeline.predict(X_test)
print(classification_report(y_test, y_pred_lr, target_names=['Not Spam', 'Spam']))

# --- Support Vector Machine (SVM) Model ---
print("\n--- Training and evaluating Support Vector Machine (SVM) ---")
svm_pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words=stopwords.words('english'))),
    ('svm', SVC(kernel='linear', C=1.0, random_state=42))
])
svm_pipeline.fit(X_train, y_train)
y_pred_svm = svm_pipeline.predict(X_test)
print(classification_report(y_test, y_pred_svm, target_names=['Not Spam', 'Spam']))
```

```
Data split complete.

--- Training and evaluating Logistic Regression ---
              precision    recall  f1-score   support

    Not Spam       0.86      0.98      0.91       165
        Spam       0.96      0.81      0.88       134

    accuracy                           0.90       299
   macro avg       0.91      0.89      0.90       299
weighted avg       0.91      0.90      0.90       299


--- Training and evaluating Support Vector Machine (SVM) ---
              precision    recall  f1-score   support

    Not Spam       0.88      0.98      0.92       165
        Spam       0.97      0.83      0.89       134

    accuracy                           0.91       299
   macro avg       0.92      0.90      0.91       299
weighted avg       0.92      0.91      0.91       299

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```python
#Using the code for the preprocessing
import pandas as pd
import re
import string
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

# Download necessary NLTK data for lemmatization and stopwords
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')

# Assume 'df' is your DataFrame from the previous steps
# If not, you must load it here
# For example: df = pd.read_csv('youtubeSpamCollection.csv')

# --- Advanced Data Cleaning and Preprocessing with Lemmatization ---
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def advanced_clean_text(text):
    text = text.lower()  # Lowercase the text
    text = re.sub(r'https?://\S+|www\.\S+', '', text)  # Remove URLs
    text = re.sub(f'[{re.escape(string.punctuation)}]', '', text)  # Remove punctuation
```

```
    # Tokenize and remove stopwords
    tokens = [word for word in text.split() if word not in stop_words]

    # Lemmatize each word
    lemmatized_tokens = [lemmatizer.lemmatize(word) for word in tokens]

    return ' '.join(lemmatized_tokens)

# Apply the new, advanced cleaning function
df['advanced_cleaned_content'] = df['CONTENT'].apply(advanced_clean_text)
print("\nDataFrame after advanced cleaning:")
print(df[['CONTENT', 'advanced_cleaned_content']].head())

# --- Re-train a model on the new data ---
X_new = df['advanced_cleaned_content']
y = df['CLASS']

X_train_new, X_test_new, y_train, y_test = train_test_split(X_new, y, test_size=0.2, random_state=42)

# Use the Logistic Regression pipeline from the previous step
lr_pipeline = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('lr', LogisticRegression(random_state=42, max_iter=1000))
])

# Train and evaluate the model on the new, improved data
print("\nTraining and evaluating Logistic Regression on advanced cleaned data...")
lr_pipeline.fit(X_train_new, y_train)

y_pred_new = lr_pipeline.predict(X_test_new)
print(classification_report(y_test, y_pred_new, target_names=['Not Spam', 'Spam']))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...

DataFrame after advanced cleaning:
                                             CONTENT  \
0  Huh, anyway check out this you[tube] channel: ...
1  Hey guys check out my new channel and our firs...
2             just for test I have to say murdev.com
```

	advanced_cleaned_content
0	huh anyway check youtube channel kobyoshi02
1	hey guy check new channel first vid u monkey i...
2	test say murdevcom

```
Training and evaluating Logistic Regression on advanced cleaned data...
              precision    recall  f1-score   support

    Not Spam       0.86      0.97      0.91       165
        Spam       0.96      0.80      0.87       134

    accuracy                           0.89       299
   macro avg       0.91      0.88      0.89       299
weighted avg       0.90      0.89      0.89       299
```