


```
from google.colab import drive
drive.mount('/content/drive')
```

 Mounted at /content/drive

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding, LSTM, Conv1D, MaxPooling1D, GlobalMaxPooling1D, Dense, Dropout
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.preprocessing import LabelEncoder

# Load and preprocess the data
data = pd.read_csv('/content/drive/MyDrive/Naive/training_data (2).csv')

# Check the column names
print(data.columns)

# Combine symptom columns into a single text column
data['text'] = data[['Symptom_1', 'Symptom_2', 'Symptom_3']].apply(lambda x: ' '.join(x.dropna().astype(str)), axis=1)

# Encode the labels to integers
label_encoder = LabelEncoder()
data['label'] = label_encoder.fit_transform(data['Disease'])

# Extract features and labels
X = data['text'].tolist()
y = data['label'].tolist()

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

# Tokenize the text data
tokenizer = Tokenizer()
tokenizer.fit_on_texts(X_train)
X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)

# Pad the sequences
max_len = 512 # You can change this value based on your data
X_train_padded = pad_sequences(X_train_seq, maxlen=max_len, padding='post')
X_test_padded = pad_sequences(X_test_seq, maxlen=max_len, padding='post')

# Build the LSTM-CNN model
input_layer = Input(shape=(max_len,))
embedding_layer = Embedding(input_dim=len(tokenizer.word_index) + 1, output_dim=128, input_length=max_len)(input_layer)
lstm_layer = LSTM(128, return_sequences=True)(embedding_layer)
conv_layer = Conv1D(64, kernel_size=3, activation='relu')(lstm_layer)
max_pooling_layer = MaxPooling1D(pool_size=2)(conv_layer)
global_max_pooling_layer = GlobalMaxPooling1D()(max_pooling_layer)
dropout_layer = Dropout(0.5)(global_max_pooling_layer)
output_layer = Dense(len(set(y)), activation='softmax')(dropout_layer)

model = Model(inputs=input_layer, outputs=output_layer)
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(X_train_padded, np.array(y_train), epochs=3, batch_size=8, validation_split=0.2)

# Evaluate the model
y_pred_probs = model.predict(X_test_padded)
y_pred = np.argmax(y_pred_probs, axis=1)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')
```

[Show hidden output](#)