```
from keras.preprocessing.text import Tokenizer
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from keras.models import Sequential
from keras.layers import Dense, Flatten
from keras.layers import Embedding
from keras.preprocessing.text import Tokenizer
from keras.utils import pad_sequences
from keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df = pd.read_csv("complete_cleaned_data CSV.csv")
```

```
label = LabelEncoder()
a = label.fit_transform(df['Label'])
df['num_label'] = a
df = df.sample(frac=1)
```

```
x=df['Text']
y=df['num_label']
```

```
#Neural Network
max_words = 20000
max_length = 500
tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(x)
sequences = tokenizer.texts_to_sequences(x)
```

Saved successfully!

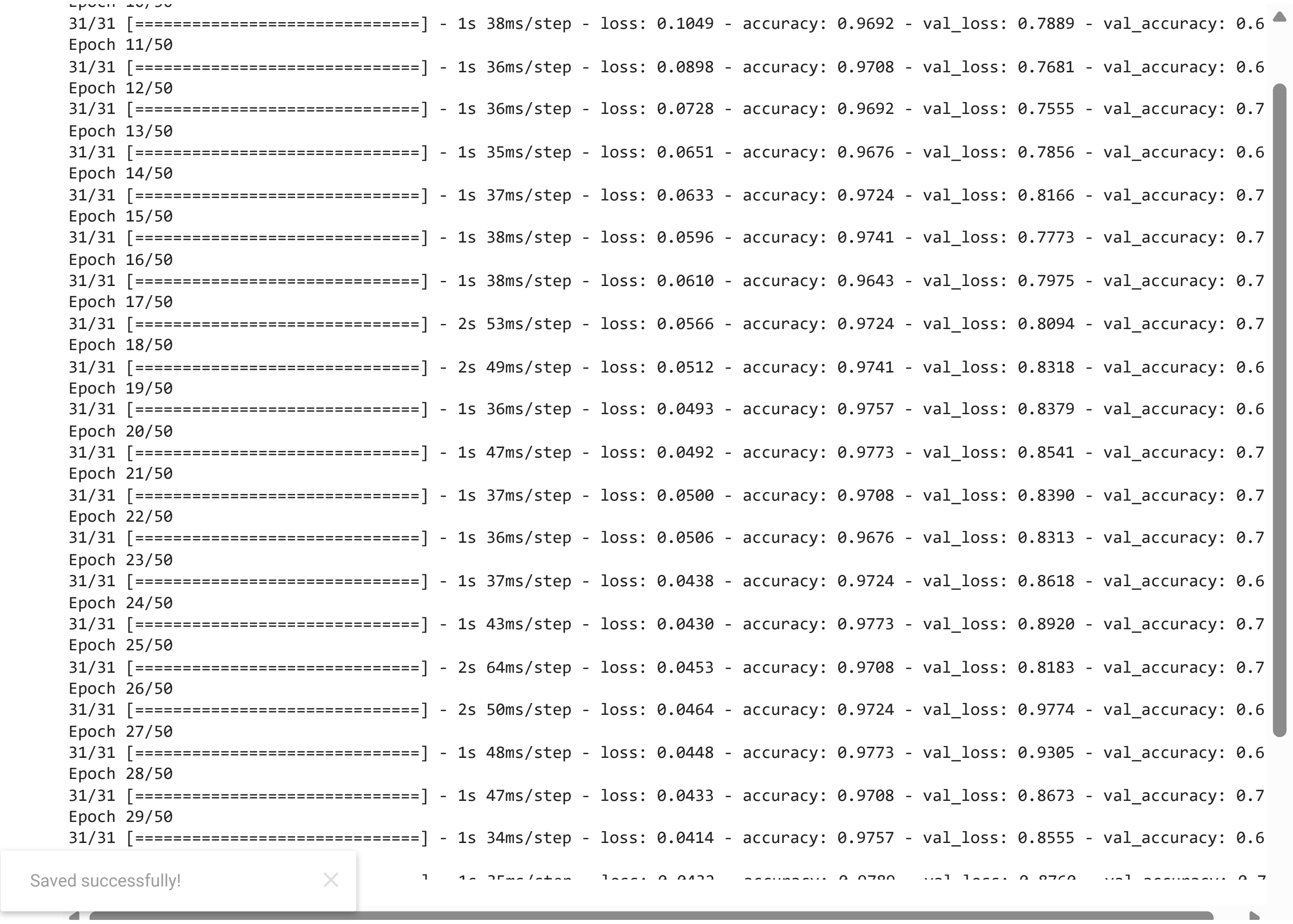
```
n=max_length)
```

```
#Building the Neural Network
model = Sequential()
model.add(Embedding(max_words, 32, input_length=max_length))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(4, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_8 (Embedding)     (None, 500, 32)           640000

 flatten_8 (Flatten)         (None, 16000)             0

 dense_15 (Dense)            (None, 128)               2048128

 dense_16 (Dense)            (None, 4)                 516

=================================================================
Total params: 2,688,644
Trainable params: 2,688,644
Non-trainable params: 0
_________________________________________________________________
```

```
#Fitting the Neural Network
hist = model.fit(x1, to_categorical(y), validation_split=0.2, epochs=50, batch_size=20)
```

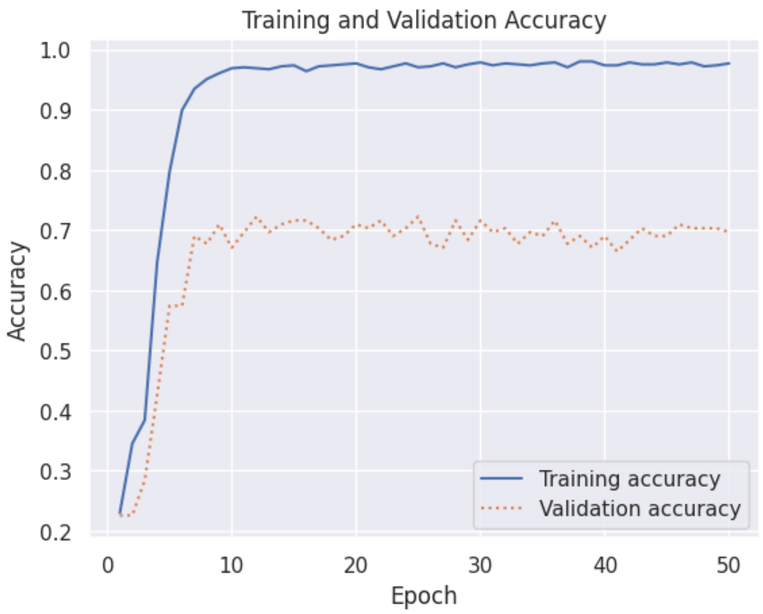
```
Epoch 3/50
31/31 [==============================] - 1s 38ms/step - loss: 1.2902 - accuracy: 0.3841 - val_loss: 1.3055 - val_accuracy: 0.2
Epoch 4/50
31/31 [==============================] - 1s 36ms/step - loss: 1.0476 - accuracy: 0.6467 - val_loss: 1.1621 - val_accuracy: 0.4
Epoch 5/50
31/31 [==============================] - 1s 37ms/step - loss: 0.7657 - accuracy: 0.7974 - val_loss: 1.0265 - val_accuracy: 0.5
Epoch 6/50
31/31 [==============================] - 1s 45ms/step - loss: 0.5372 - accuracy: 0.8995 - val_loss: 0.9451 - val_accuracy: 0.5
Epoch 7/50
31/31 [==============================] - 2s 54ms/step - loss: 0.3320 - accuracy: 0.9352 - val_loss: 0.8167 - val_accuracy: 0.6
```



```
Epoch 10/50
31/31 [==============================] - 1s 38ms/step - loss: 0.1049 - accuracy: 0.9692 - val_loss: 0.7889 - val_accuracy: 0.6
Epoch 11/50
31/31 [==============================] - 1s 36ms/step - loss: 0.0898 - accuracy: 0.9708 - val_loss: 0.7681 - val_accuracy: 0.6
Epoch 12/50
31/31 [==============================] - 1s 36ms/step - loss: 0.0728 - accuracy: 0.9692 - val_loss: 0.7555 - val_accuracy: 0.7
Epoch 13/50
31/31 [==============================] - 1s 35ms/step - loss: 0.0651 - accuracy: 0.9676 - val_loss: 0.7856 - val_accuracy: 0.6
Epoch 14/50
31/31 [==============================] - 1s 37ms/step - loss: 0.0633 - accuracy: 0.9724 - val_loss: 0.8166 - val_accuracy: 0.7
Epoch 15/50
31/31 [==============================] - 1s 38ms/step - loss: 0.0596 - accuracy: 0.9741 - val_loss: 0.7773 - val_accuracy: 0.7
Epoch 16/50
31/31 [==============================] - 1s 38ms/step - loss: 0.0610 - accuracy: 0.9643 - val_loss: 0.7975 - val_accuracy: 0.7
Epoch 17/50
31/31 [==============================] - 2s 53ms/step - loss: 0.0566 - accuracy: 0.9724 - val_loss: 0.8094 - val_accuracy: 0.7
Epoch 18/50
31/31 [==============================] - 2s 49ms/step - loss: 0.0512 - accuracy: 0.9741 - val_loss: 0.8318 - val_accuracy: 0.6
Epoch 19/50
31/31 [==============================] - 1s 36ms/step - loss: 0.0493 - accuracy: 0.9757 - val_loss: 0.8379 - val_accuracy: 0.6
Epoch 20/50
31/31 [==============================] - 1s 47ms/step - loss: 0.0492 - accuracy: 0.9773 - val_loss: 0.8541 - val_accuracy: 0.7
Epoch 21/50
31/31 [==============================] - 1s 37ms/step - loss: 0.0500 - accuracy: 0.9708 - val_loss: 0.8390 - val_accuracy: 0.7
Epoch 22/50
31/31 [==============================] - 1s 36ms/step - loss: 0.0506 - accuracy: 0.9676 - val_loss: 0.8313 - val_accuracy: 0.7
Epoch 23/50
31/31 [==============================] - 1s 37ms/step - loss: 0.0438 - accuracy: 0.9724 - val_loss: 0.8618 - val_accuracy: 0.6
Epoch 24/50
31/31 [==============================] - 1s 43ms/step - loss: 0.0430 - accuracy: 0.9773 - val_loss: 0.8920 - val_accuracy: 0.7
Epoch 25/50
31/31 [==============================] - 2s 64ms/step - loss: 0.0453 - accuracy: 0.9708 - val_loss: 0.8183 - val_accuracy: 0.7
Epoch 26/50
31/31 [==============================] - 2s 50ms/step - loss: 0.0464 - accuracy: 0.9724 - val_loss: 0.9774 - val_accuracy: 0.6
Epoch 27/50
31/31 [==============================] - 1s 48ms/step - loss: 0.0448 - accuracy: 0.9773 - val_loss: 0.9305 - val_accuracy: 0.6
Epoch 28/50
31/31 [==============================] - 1s 47ms/step - loss: 0.0433 - accuracy: 0.9708 - val_loss: 0.8673 - val_accuracy: 0.7
Epoch 29/50
31/31 [==============================] - 1s 34ms/step - loss: 0.0414 - accuracy: 0.9757 - val_loss: 0.8555 - val_accuracy: 0.6
                              ] - 1s 35ms/step - loss: 0.0422 - accuracy: 0.9789 - val_loss: 0.8760 - val_accuracy: 0.7
```

```
#Plotting
sns.set()

acc = hist.history['accuracy']
val = hist.history['val_accuracy']
epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, '-', label='Training accuracy')
plt.plot(epochs, val, ':', label='Validation accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.plot()
```

```
    []
```



Colab paid products - Cancel contracts here

✓ 0s completed at 12:06 AM ● ×

Saved successfully! ×