

NIM : 2440016804

Nama : Rio Planiana

Kelas : LA01

Nomor 2a, 2b, 2c, dan 2d

2a. Preprocess & prepare dataset

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns

Reading and looking at the dataset

In [2]: # Importing the dataset, downloaded file is in the same folder
csv_path = "clickbait.csv"
clickbait_df = pd.read_csv(csv_path)

In [3]: clickbait_df.head(5)

Out[3]:
```

	headline	clickbait
0	Should I Get Bings	1
1	Which TV Female Friend Group Do You Belong In	1
2	The New 'Star Wars: The Force Awakens' Trailer...	1
3	This Vine Of New York On 'Celebrity Big Brother...	1
4	A Couple Did A Stunning Photo Shoot With Their...	1

```
In [4]: clickbait_df.info()

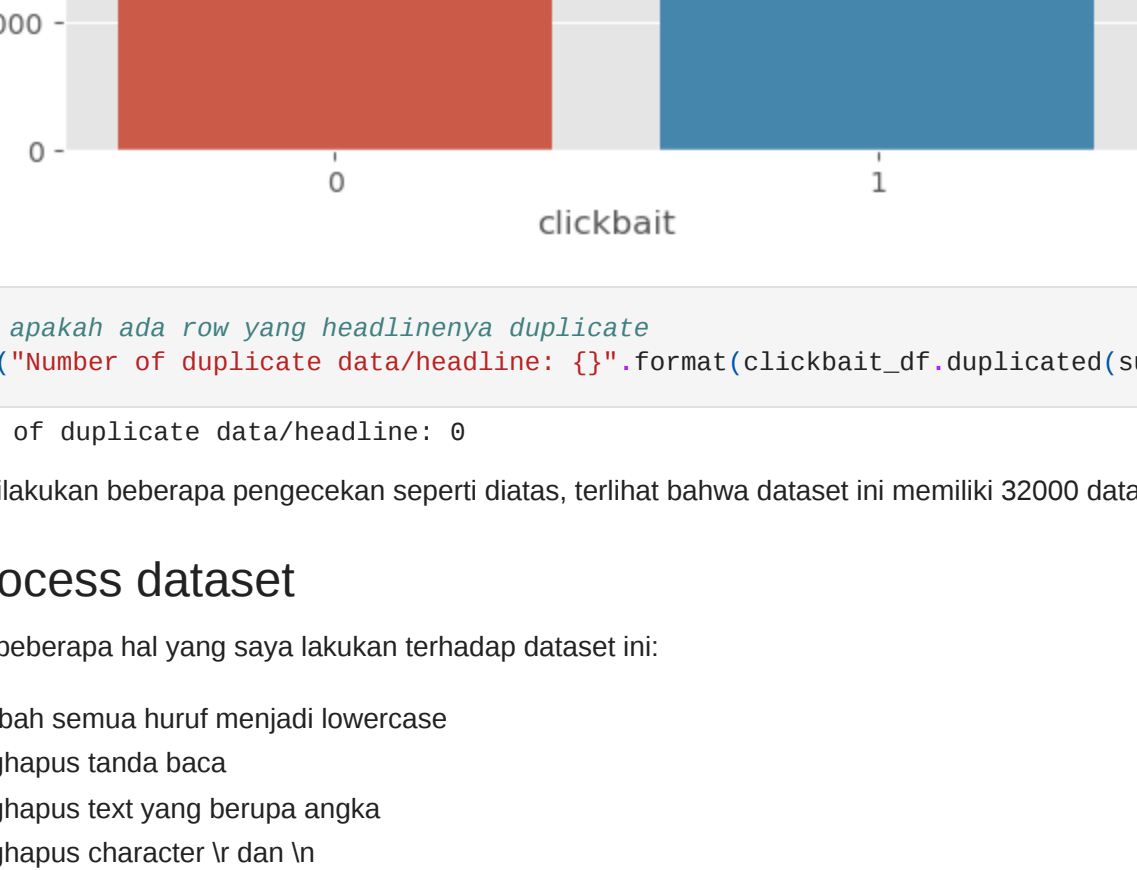
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32000 entries, 0 to 31999
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype  
---  --
 0   headline    32000 non-null  object  
 1   clickbait    32000 non-null  int64   
dtypes: int64(1), object(1)
memory usage: 500.1+ KB

In [5]: clickbait_df.isnull().sum()

Out[5]:
headline      0
clickbait      0
dtype: int64

In [6]: #set style ggplot
style.use('ggplot')

In [7]: plt.figure(figsize=(7,7))
sns.countplot(x='clickbait', data=clickbait_df)
plt.title('Clickbait Class Distribution')
plt.show()
```



```
In [8]: # Cek apakah ada row yang headlinenya duplicate
print(NumberOf duplicate data/headline: {}").format(clickbait_df.duplicated(subset=["headline"]).sum())

Number of duplicate data/headline: 0
```

Setelah dilakukan beberapa pengecekan seperti diatas, terlihat bahwa dataset ini memiliki 32000 data dengan 50% class 1 dan 50% class 0. Tidak ada data yang kosong/duplikat

Preprocess dataset

Terdapat beberapa hal yang saya lakukan terhadap dataset ini:

1. Merubah semua huruf menjadi lowercase
 2. Menghapus tanda baca
 3. Menghapus text yang berupa angka
 4. Menghapus character \r dan \n
 5. Menghapus stopwords dalam bahasa Inggris
 6. Melakukan lemmatization
 7. Melakukan tokenization
 8. Melakukan padding untuk menyesuaikan input dengan model
 9. Menghapus spasi yang berlebihan
1. Seluruh string yang ada pada kolom headline perlu diubah menjadi lowercase agar dimensionalitas pada data berkurang dan semua kata yang sama akan dianggap sama walaupun penggunaan huruf kapitalnya berbeda (misalnya kata 'learning', 'Learning', dan 'LeArNing' akan dianggap sebagai kata yang sama)
2. Menghapus tanda baca pada text juga merupakan step yang penting karena tanda-tanda baca tidak memiliki makna semantik yang signifikan untuk menentukan apakah sebuah headline merupakan clickbait atau tidak. Tidak hanya tanda baca ini tidak membawa hal positif, tanda baca ini justru bisa berdampak negatif karena akan memengaruhi bagaimana model belajar mengklasifikasi headline mana yang merupakan clickbait berdasarkan kata yang digunakan
3. Menghapus angka juga merupakan step yang penting seperti step nomor 2. Angka pada headline tidak mengandung makna yang signifikan
4. Menghapus \r dan \n juga membantu data text menjadi lebih bersih (menghilangkan characters yang tidak terlalu bermakna)
5. Menghapus stopwords juga menjadi step yang penting karena biasanya dalam bahasa Inggris terdapat beberapa kata yang hampir selalu digunakan dalam setiap kalimat (seperti "the" dan "and") sehingga kata-kata tersebut tidak membawa makna apa-apa dalam menentukan clickbait
6. Lemmatization membantu mengurangi dimensionalitas data dan membantu model dalam memahami konteks dari sebuah kata dengan lebih baik. Lemmatization berfungsi dengan cara merubah sebuah kata ke base formnya (misalnya 'studied' menjadi 'study') tanpa menghilangkan makna dari katanya (berbeda dengan stemming yang akan merubah kata 'studied' menjadi 'study' yang bukannya merupakan kata dalam bahasa Inggris)
7. Tokenization berguna untuk model yang akan di-train dengan cara merubah setiap kata pada text menjadi integer-integer yang unik sehingga sebuah text bisa direpresentasikan sebagai sequence yang akan digunakan sebagai input model
8. Padding dilakukan agar setiap text memiliki shape yang sama karena model akan mengharapkan input dengan shape tertentu (tidak bervariasi). Padding dilakukan dengan menambahkan 0 pada sequence text sampai semua sequence memiliki length yang sama
9. Penghapusan space di akhir preprocessing sangat berguna untuk menghapus karakter spasi yang berlebihan, terutama akibat step preprocessing yang dilakukan sebelumnya

```
In [9]: def remove_unnecessary_spaces(string):
string = string.strip() # remove spaces di depan dan belakang string
string = ' '.join(string.split()) # remove unnecessary spaces antar kata di string
return string

In [10]: import string as st
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

def clean_text_before_tokenize(original_df, column):
    # copy the dataframe so we do not change the original dataframe
    df = original_df.copy()
    # turn all text to lowercase
    df[column] = df[column].str.lower() # convert to lowercase
    # remove durations
    df[column] = df[column].str.replace('\d+', '') # re.escape(st.punctuation), '', x))
    # remove numbers
    df[column] = df[column].str.replace('\d+', '') # remove numbers
    # remove \r & \n
    df[column] = df[column].str.replace('\r', '')
    df[column] = df[column].str.replace('\n', '')
    # remove stopwords
    stop_words = set(stopwords.words('english'))
    df[column] = df[column].apply(lambda x: ' '.join([word for word in word_tokenize(x) if word.lower() not in stop_words]))
    # remove all unnecessary/extra spaces
    df[column] = df[column].apply(remove_unnecessary_spaces)

    return df

def lemmatize_text(text):
    tokens = word_tokenize(text)
    lemmatizer = WordNetLemmatizer()

    return [lemmatizer.lemmatize(token) for token in tokens]

def apply_lemmatization(data):
    data = [lemmatize_text(text) for text in data.tolist()]

    return data

def tokenize_data_for_LSTM(X_train, X_val, X_test, vocab_size, maxlen):
    X_train = apply_lemmatization(X_train)
    X_val = apply_lemmatization(X_val)
    X_test = apply_lemmatization(X_test)
    texts = X_train + X_val + X_test
    tokenizer = Tokenizer(num_words = vocab_size)
    tokenizer.fit_on_texts(texts)
    X_train = tokenizer.texts_to_sequences(X_train)
    X_val = tokenizer.texts_to_sequences(X_val)
    X_test = tokenizer.texts_to_sequences(X_test)
    # pad the sequences to make all of them the same length
    X_train = pad_sequences(X_train, maxlen = maxlen)
    X_val = pad_sequences(X_val, maxlen = maxlen)
    X_test = pad_sequences(X_test, maxlen = maxlen)

    return X_train, X_val, X_test
```

Functions dibawah ini adalah helper functions yang membantu untuk menentukan nilai vocab_size dan maxlen. Vocab_size baiknya memiliki value sebanyak jumlah kata yang unik pada dataset, sedangkan maxlen baiknya memiliki value sebanyak panjang text terpanjang pada dataset

```
In [11]: def get_num_unique_words(df, column):
all_words = []
for words in df[column].values:
    all_words.extend(words)
unique_words = set(all_words)

return len(unique_words)

def get_len_longest_string(df, column):
return len(max(df[column], key=len))

In [12]: def preprocess_dataset_LSTM(original_df, column):
df = original_df.copy()
df = clean_text_before_tokenize(df, column)

return df
```

Preprocess dataframe (sebelum tokenization)

```
In [13]: cleaned_clickbait_df = preprocess_dataset_LSTM(clickbait_df, 'headline')
cleaned_clickbait_df.head(5)

C:\Users\User\AppData\Local\Temp\ipykernel_6104\2926137536.py:17: FutureWarning: The default value of regex will change from True to False in a future version.
df[column] = df[column].str.replace('\d+', '') # remove numbers

Out[13]:
```

	headline	clickbait
0	get bings	1
1	tv female friend group belong	1
2	new star wars force awakens trailer give chills	1
3	vine new york celebrity big brother fucking pe...	1
4	couple stunning photo shoot baby learning insp...	1

Split dataset

Dataset saya split menjadi 80/10/10

```
In [14]: from sklearn.model_selection import train_test_split

X = cleaned_clickbait_df['headline'].values
y = cleaned_clickbait_df['clickbait'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
X_train, X_val, y_train, y_val = train_test_split(X_test, y_test, test_size = 0.5, random_state = 42)

print(X_train.shape, X_val.shape, X_test.shape, y_train.shape, y_val.shape, y_test.shape)

(25600,) (3200,) (3200,) (25600,) (3200,) (3200,)
```

Function dibawah ini menyelesaikan preparation dataset untuk model yang akan saya train dengan melakukan lemmatization dan tokenization serta padding

```
In [15]: def prepare_inputs_LSTM(df, column, X_train, X_val, X_test):
vocab_size = get_num_unique_words(df, column) * 5
maxlen = get_len_longest_string(df, column) + 1
print("Vocab size is {} and max length is {}".format(vocab_size, maxlen))
X_train, X_val, X_test = tokenize_data_for_LSTM(X_train, X_val, X_test, vocab_size, maxlen)

return X_train, X_val, X_test, vocab_size, maxlen

In [16]: X_train, X_val, X_test, vocab_size, maxlen = prepare_inputs_LSTM(cleaned_clickbait_df, 'headline', X_train, X_val, X_test)

Vocab size is 71 and max length is 127
```

2b. Membuat model LSTM

Untuk membuat model yang dapat melakukan clickbait text classification, saya memilih untuk menggunakan LSTM (Long-Short Term Memory)

LSTM merupakan salah satu jenis RNN yang dapat meng-handle sequences of data, salah satu data yang bisa di-handle adalah text. LSTM mampu mendapatkan konteks dan ketergantungan antar kata pada sebuah kalimat. LSTM memiliki memory cells yang dapat menyimpan informasi untuk waktu yang sangat lama sehingga dapat digunakan untuk mendapatkan pola sequential yang ada pada teks dan menghasilkan prediksi yang lebih baik berdasarkan pola tersebut, tidak seperti jenis RNN lainnya.

Karakteristik dan kelebihan LSTM diatas sangat cocok dengan text classification berupa clickbait seperti pada kasus ini. Untuk mengetahui apakah sebuah headline itu clickbait atau bukan, kita biasanya harus melihat konteks kalimatnya secara keseluruhan. Hal tersebut juga biasanya berarti kita harus mempertimbangkan penggunaan kata dan urutan penggunaan kata tersebut. Contohnya, headline 'President of X ate chickens' mungkin terdengar normal dan bukan clickbait, tetapi headline 'Chickens ate President of X' akan terdengar seperti sebuah headline yang tidak mungkin terjadi dan bisa dikategorikan sebagai clickbait, padahal kata yang digunakan sama. LSTM mampu melihat konteks sebuah kalimat berdasarkan sequence kata yang digunakan sehingga LSTM dapat mengatasi problem tersebut dan menghasilkan klasifikasi yang baik

Selain itu, LSTM juga memiliki gates yang mampu mengendalikan flow dari informasi yang ada pada networknya sehingga LSTM dapat mengatasi vanishing gradient problem dengan lebih baik jika dibandingkan dengan tipe RNN lainnya

```
In [17]: from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout, GlobalMaxPooling1D
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

embedding_size = 32

model = Sequential()
model.add(Embedding(vocab_size, embedding_size, input_length=maxlen))
model.add(LSTM(64, return_sequences=True))
model.add(LSTM(32, return_sequences=False))
model.add(Dropout(0.3))
model.add(Dense(1, activation='sigmoid'))
model.summary()

Model: "sequential"

Layer (type) Output Shape Param #
-----
embedding (Embedding) (None, 127, 32) 2272
lstm (LSTM) (None, 127, 64) 24832
lstm_1 (LSTM) (None, 32) 12416
dropout (Dropout) (None, 32) 0
dense (Dense) (None, 1) 0

Total params: 39,553
Trainable params: 39,553
Non-trainable params: 0

Model yang saya buat diatas memiliki sebuah embedding layer yang akan menerima input text (yang sudah diubah menjadi integer) dan akan mengubahnya menjadi embeddings yang akan membantu model mempelajari makna dari setiap kata dan hubungan antar kata pada setiap text
```

Kemudian, terdapat 2 LSTM layer (layer kedua return_sequencesnya False) yang akan menghandle data-data dalam bentuk sequence. LSTM layer berfungsi untuk menyimpan informasi-informasi penting pada sequence input dan mengabkan informasi yang dianggap tidak relevan melalui hidden state yang akan memberikan informasi dari langkah pertama ke langkah selanjutnya

Dropout layer ditambahkan untuk menghindari overfitting

Training model LSTM

```
In [18]: callbacks = [
    EarlyStopping(
        monitor='val_loss',
        min_delta=1e-4,
        patience=5,
        verbose=1
    ),
    ModelCheckpoint(
        filepath='weights_h5',
        monitor='val_loss',
        mode='min',
        save_best_only=True,
        save_weights_only=True,
        verbose=1
    )
]

Model akan di-train dengan max epochs = 40 (training akan di-stop ketika validation loss tidak kunjung membaik dengan patience = 5). Initial_epoch saya set menjadi 5 agar training tidak berhenti terlalu awal
```

```
In [19]: from tensorflow.keras.optimizers import Adam
batch_size = 256
epochs = 40

model.compile(loss='binary_crossentropy', optimizer=Adam(learning_rate=0.01), metrics=['accuracy'])
history = model.fit(X_train, y_train, batch_size=batch_size, validation_data=(X_val, y_val), epochs=epochs, callbacks=callbacks,
                    initial_epoch = 5)

Epoch 6/40 =====
Epoch 6: val_loss improved from inf to 0.44510, saving model to weights_h5
100/100 [=====] - ETA: 0s - loss: 0.4770 - accuracy: 0.7561
Epoch 7/40 =====
Epoch 7: val_loss improved from 0.44510 to 0.44047, saving model to weights_h5
100/100 [=====] - ETA: 0s - loss: 0.4480 - accuracy: 0.7850
Epoch 8/40 =====
Epoch 8: val_loss improved from 0.44047 to 0.44047, saving model to weights_h5
100/100 [=====] - ETA: 0s - loss: 0.4427 - accuracy: 0.7832
Epoch 9/40 =====
Epoch 9: val_loss did not improve from 0.44047
100/100 [=====] - ETA: 0s - loss: 0.4377 - accuracy: 0.7868
Epoch 10/40 =====
Epoch 10: val_loss did not improve from 0.44047
100/100 [=====] - ETA: 0s - loss: 0.4345 - accuracy: 0.7869
Epoch 11/40 =====
Epoch 11: val_loss did not improve from 0.44047
100/100 [=====] - ETA: 0s - loss: 0.4331 - accuracy: 0.7869
Epoch 12/40 =====
Epoch 12: val_loss did not improve from 0.44047
100/100 [=====] - ETA: 0s - loss: 0.4327 - accuracy: 0.7859
Epoch 13/40 =====
Epoch 13: val_loss improved from 0.44047 to 0.44030, saving model to weights_h5
100/100 [=====] - ETA: 0s - loss: 0.4324 - accuracy: 0.7873
Epoch 14/40 =====
Epoch 14: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4312 - accuracy: 0.7867
Epoch 15/40 =====
Epoch 15: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4306 - accuracy: 0.7874
Epoch 16/40 =====
Epoch 16: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4296 - accuracy: 0.7867
Epoch 17/40 =====
Epoch 17: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4296 - accuracy: 0.7867
Epoch 18/40 =====
Epoch 18: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 19/40 =====
Epoch 19: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 20/40 =====
Epoch 20: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 21/40 =====
Epoch 21: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 22/40 =====
Epoch 22: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 23/40 =====
Epoch 23: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 24/40 =====
Epoch 24: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 25/40 =====
Epoch 25: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 26/40 =====
Epoch 26: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 27/40 =====
Epoch 27: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 28/40 =====
Epoch 28: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 29/40 =====
Epoch 29: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 30/40 =====
Epoch 30: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 31/40 =====
Epoch 31: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 32/40 =====
Epoch 32: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 33/40 =====
Epoch 33: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 34/40 =====
Epoch 34: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 35/40 =====
Epoch 35: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 36/40 =====
Epoch 36: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 37/40 =====
Epoch 37: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 38/40 =====
Epoch 38: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 39/40 =====
Epoch 39: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 40/40 =====
Epoch 40: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 41/40 =====
Epoch 41: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 42/40 =====
Epoch 42: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 43/40 =====
Epoch 43: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 44/40 =====
Epoch 44: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 45/40 =====
Epoch 45: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 46/40 =====
Epoch 46: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 47/40 =====
Epoch 47: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 48/40 =====
Epoch 48: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 49/40 =====
Epoch 49: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 50/40 =====
Epoch 50: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 51/40 =====
Epoch 51: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 52/40 =====
Epoch 52: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 53/40 =====
Epoch 53: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 54/40 =====
Epoch 54: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 55/40 =====
Epoch 55: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 56/40 =====
Epoch 56: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 57/40 =====
Epoch 57: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 58/40 =====
Epoch 58: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 59/40 =====
Epoch 59: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 60/40 =====
Epoch 60: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 61/40 =====
Epoch 61: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 62/40 =====
Epoch 62: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 63/40 =====
Epoch 63: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 64/40 =====
Epoch 64: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 65/40 =====
Epoch 65: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 66/40 =====
Epoch 66: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 67/40 =====
Epoch 67: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 68/40 =====
Epoch 68: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 69/40 =====
Epoch 69: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 70/40 =====
Epoch 70: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 71/40 =====
Epoch 71: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 72/40 =====
Epoch 72: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 73/40 =====
Epoch 73: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 74/40 =====
Epoch 74: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 75/40 =====
Epoch 75: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 76/40 =====
Epoch 76: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 77/40 =====
Epoch 77: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 78/40 =====
Epoch 78: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 79/40 =====
Epoch 79: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 80/40 =====
Epoch 80: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 81/40 =====
Epoch 81: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 82/40 =====
Epoch 82: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 83/40 =====
Epoch 83: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 84/40 =====
Epoch 84: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 85/40 =====
Epoch 85: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 86/40 =====
Epoch 86: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 87/40 =====
Epoch 87: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 88/40 =====
Epoch 88: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 89/40 =====
Epoch 89: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 90/40 =====
Epoch 90: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 91/40 =====
Epoch 91: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 92/40 =====
Epoch 92: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 93/40 =====
Epoch 93: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 94/40 =====
Epoch 94: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 95/40 =====
Epoch 95: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 96/40 =====
Epoch 96: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 97/40 =====
Epoch 97: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 98/40 =====
Epoch 98: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 99/40 =====
Epoch 99: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 100/40 =====
Epoch 100: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 101/40 =====
Epoch 101: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 102/40 =====
Epoch 102: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 103/40 =====
Epoch 103: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 104/40 =====
Epoch 104: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 105/40 =====
Epoch 105: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 106/40 =====
Epoch 106: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 107/40 =====
Epoch 107: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 108/40 =====
Epoch 108: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 109/40 =====
Epoch 109: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 110/40 =====
Epoch 110: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 111/40 =====
Epoch 111: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 112/40 =====
Epoch 112: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 113/40 =====
Epoch 113: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 114/40 =====
Epoch 114: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 115/40 =====
Epoch 115: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 116/40 =====
Epoch 116: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 117/40 =====
Epoch 117: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 118/40 =====
Epoch 118: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 119/40 =====
Epoch 119: val_loss did not improve from 0.44029
100/100 [=====] - ETA: 0s - loss: 0.4289 - accuracy: 0.7882
Epoch 120
```