```
# Imports
import pandas as pd
import re
from collections import Counter
import matplotlib.pyplot as plt
import seaborn as sns
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import nltk

nltk.download('punkt')
nltk.download('stopwords')

from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split

!pip install tensorflow
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping



# Load the text file
df = pd.read_csv('yelp_labelled.txt', sep='\t', names=['Review',
'Sentiment'])

# Unusual Characters
unusual_chars = []
for review in df['Review']:
    unusual_chars.extend([char for char in review if not
char.isalnum() and not char.isspace()])
unusual_chars_count = Counter(unusual_chars)

print("Unusual Characters:")
print(unusual_chars_count)

# Vocabulary Size
words = ' '.join(df['Review']).lower()
words = re.sub(r'[^a-zA-Z\s]', '', words)
words = word_tokenize(words)
unique_words = set(words)
vocabulary_size = len(unique_words)
print("\nVocabulary Size:", vocabulary_size)

# Word Frequency Distribution
word_freq = Counter(words)
```

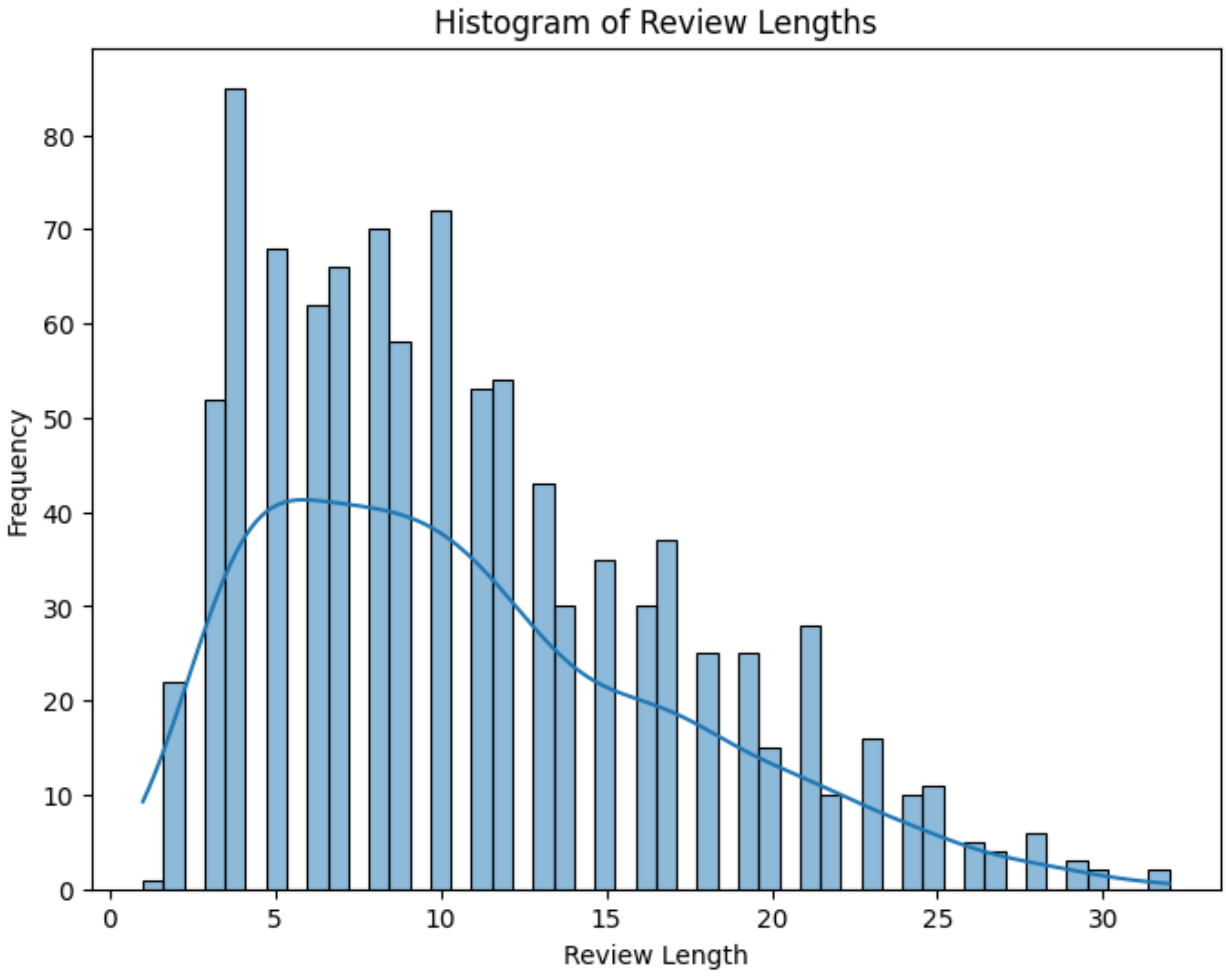
```
most_common_words = word_freq.most_common(20)
print("\nMost Common Words:")
print(most_common_words)

# Statistical Justification
review_lengths = df['Review'].apply(lambda x: len(x.split()))
max_seq_length = review_lengths.quantile(0.95)
print("\nMax Sequence Length (95th percentile):", max_seq_length)

# Data Cleaning

# Convert text to lowercase
df['Review'] = df['Review'].str.lower()

# Remove punctuation
df['Review'] = df['Review'].apply(lambda x: re.sub(r'[^\w\s]', '', x))

# Remove stopwords
stop_words = set(stopwords.words('english'))
df['Review'] = df['Review'].apply(lambda x: ' '.join([word for word in
x.split() if word not in stop_words]))

# Handling Null Values
df.dropna(inplace=True)

# Displaying Histogram of Review Lengths
plt.figure(figsize=(8, 6))
sns.histplot(review_lengths, bins=50, kde=True)
plt.title('Histogram of Review Lengths')
plt.xlabel('Review Length')
plt.ylabel('Frequency')
plt.show()
```

```
[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!

Requirement already satisfied: tensorflow in
/usr/local/lib/python3.10/dist-packages (2.15.0)
Requirement already satisfied: absl-py>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (23.5.26)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1
in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
```

```
Requirement already satisfied: h5py>=2.9.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (16.0.6)
Requirement already satisfied: ml-dtypes~=0.2.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.23.5)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (23.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!
=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.20.3)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (4.5.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.34.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.60.0)
Requirement already satisfied: tensorboard<2.16,>=2.15 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.1)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0
in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)
Requirement already satisfied: keras<2.16,>=2.15.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0-
>tensorflow) (0.42.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
>tensorflow) (2.17.3)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
>tensorflow) (1.2.0)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
>tensorflow) (3.5.1)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
```

```
>tensorflow) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0
in /usr/local/lib/python3.10/dist-packages (from
tensorboard<2.16,>=2.15->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
>tensorflow) (3.0.1)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.16,>=2.15->tensorflow) (5.3.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.16,>=2.15->tensorflow) (0.3.0)
Requirement already satisfied: rsa<5,>=3.1.4 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.16,>=2.15->tensorflow) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from google-auth-
oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow) (1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.16,>=2.15->tensorflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.16,>=2.15->tensorflow) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.16,>=2.15->tensorflow) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.16,>=2.15->tensorflow) (2023.11.17)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1-
>tensorboard<2.16,>=2.15->tensorflow) (2.1.3)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in
/usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1-
>google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (0.5.1)
Requirement already satisfied: oauthlib>=3.0.0 in
/usr/local/lib/python3.10/dist-packages (from requests-
oauthlib>=0.7.0->google-auth-oauthlib<2,>=0.5-
>tensorboard<2.16,>=2.15->tensorflow) (3.2.2)
Unusual Characters:
Counter({'.': 967, ',': 364, '!': 251, "'": 209, '-': 50, '"': 27,
')': 18, '(': 18, '/': 14, '&': 12, ':': 10, '$': 8, '?': 6, ';': 4,
'*': 3, '%': 2, '+': 2})

Vocabulary Size: 2050

Most Common Words:
```

```
[('the', 584), ('and', 389), ('i', 304), ('was', 295), ('a', 237),
('to', 219), ('is', 171), ('this', 143), ('it', 132), ('of', 127),
('food', 124), ('not', 118), ('for', 110), ('in', 107), ('place',
106), ('good', 95), ('service', 83), ('we', 79), ('very', 75), ('my',
73)]

Max Sequence Length (95th percentile): 23.0
```




```python
# Tokenization

# Convert reviews to a list
reviews = df['Review'].tolist()

# Tokenize the text
tokenizer = Tokenizer()
tokenizer.fit_on_texts(reviews)

# Vocabulary size after tokenization
vocab_size_after_tokenization = len(tokenizer.word_index) + 1
```

```
print("\nVocabulary Size after Tokenization:",
vocab_size_after_tokenization)

# Convert text to sequences of integers
sequences = tokenizer.texts_to_sequences(reviews)

# Statistical justification
max_sequence_length_tokenized = max(len(sequence) for sequence in
sequences)
print("Max Sequence Length after Tokenization:",
max_sequence_length_tokenized)

#B3

# Pad sequences to a uniform length
padded_sequences = pad_sequences(sequences,
maxlen=int(max_seq_length), padding='post')

# padded sequence
print("\nPadded Sequence:")
print(padded_sequences[0])
print("\nSequence Lengths after Padding:")
sequence_lengths_after_padding = [len(seq) for seq in
padded_sequences]
print(sequence_lengths_after_padding[:5])


Vocabulary Size after Tokenization: 1967
Max Sequence Length after Tokenization: 18

Padded Sequence:
[330  90   2   0   0   0   0   0   0   0   0   0   0   0   0   0
0
   0   0   0   0   0]

Sequence Lengths after Padding:
[23, 23, 23, 23, 23]

# Splitting data
train_size = 0.8
val_size = 0.1
test_size = 0.1

train_val_sequences, test_sequences, train_val_labels, test_labels =
train_test_split(
    padded_sequences, df['Sentiment'], test_size=test_size,
random_state=42)

train_sequences, val_sequences, train_labels, val_labels =
train_test_split(
    train_val_sequences, train_val_labels,
```

```
test_size=val_size/(train_size + val_size), random_state=42)

# Displaying the sizes of the splits
print(f"Training Set Size: {len(train_sequences)}")
print(f"Validation Set Size: {len(val_sequences)}")
print(f"Test Set Size: {len(test_sequences)}")
```

```
Training Set Size: 800
Validation Set Size: 100
Test Set Size: 100
```

```
# Create a new DataFrame
prepared_data = pd.DataFrame({
    'Padded_Sequence': padded_sequences.tolist(),
    'Sentiment': df['Sentiment']
})

# Save data
prepared_data.to_csv('prepared_yelp_dataset.csv', index=False)



# Define the model architecture
vocab_size = vocab_size_after_tokenization
embedding_dim = 100
max_length = int(max_seq_length)
num_classes = 2

model = Sequential()
model.add(Embedding(input_dim=vocab_size, output_dim=embedding_dim,
input_length=max_length))
model.add(LSTM(units=128))
model.add(Dense(units=64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(units=num_classes, activation='sigmoid'))

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# Display model summary
model.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_11 (Embedding)    (None, 23, 100)           196700

 lstm_11 (LSTM)              (None, 128)               117248
```

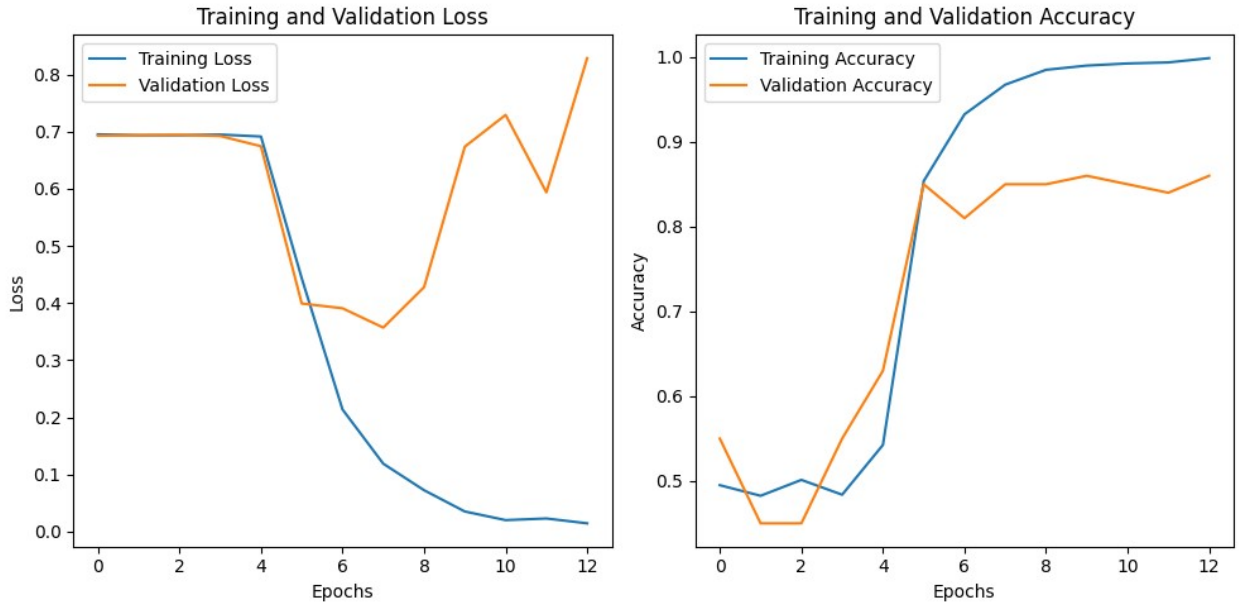
```
 dense_22 (Dense)            (None, 64)                8256

 dropout_11 (Dropout)        (None, 64)                0

 dense_23 (Dense)            (None, 2)                 130

=================================================================
Total params: 322334 (1.23 MB)
Trainable params: 322334 (1.23 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
# Define the model architecture with dropout layers (Brownlee, 2021)
model = Sequential()
model.add(Embedding(input_dim=vocab_size, output_dim=embedding_dim,
input_length=max_length))
model.add(LSTM(units=128))
model.add(Dense(units=64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(units=num_classes, activation='sigmoid'))

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# Implement early stopping
early_stop = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)

# Train the model with dropout and early stopping
history = model.fit(train_sequences, train_labels_one_hot,
                    epochs=num_epochs,
                    validation_data=(val_sequences,
val_labels_one_hot),
                    callbacks=[early_stop])
```

```
Epoch 1/20

/usr/local/lib/python3.10/dist-packages/tensorflow/python/data/ops/
structured_function.py:258: UserWarning: Even though the
`tf.config.experimental_run_functions_eagerly` option is set, this
option does not apply to tf.data functions. To force eager execution
of tf.data functions, please use
`tf.data.experimental.enable_debug_mode()`.
  warnings.warn(

25/25 [==============================] - 5s 213ms/step - loss: 0.6951
- accuracy: 0.4950 - val_loss: 0.6928 - val_accuracy: 0.5500
Epoch 2/20
```

```
25/25 [==============================] - 7s 272ms/step - loss: 0.6937
- accuracy: 0.4825 - val_loss: 0.6935 - val_accuracy: 0.4500
Epoch 3/20
25/25 [==============================] - 5s 206ms/step - loss: 0.6937
- accuracy: 0.5013 - val_loss: 0.6942 - val_accuracy: 0.4500
Epoch 4/20
25/25 [==============================] - 6s 236ms/step - loss: 0.6948
- accuracy: 0.4837 - val_loss: 0.6923 - val_accuracy: 0.5500
Epoch 5/20
25/25 [==============================] - 8s 307ms/step - loss: 0.6914
- accuracy: 0.5425 - val_loss: 0.6744 - val_accuracy: 0.6300
Epoch 6/20
25/25 [==============================] - 5s 210ms/step - loss: 0.4434
- accuracy: 0.8537 - val_loss: 0.3992 - val_accuracy: 0.8500
Epoch 7/20
25/25 [==============================] - 6s 259ms/step - loss: 0.2138
- accuracy: 0.9325 - val_loss: 0.3908 - val_accuracy: 0.8100
Epoch 8/20
25/25 [==============================] - 5s 204ms/step - loss: 0.1189
- accuracy: 0.9675 - val_loss: 0.3570 - val_accuracy: 0.8500
Epoch 9/20
25/25 [==============================] - 6s 234ms/step - loss: 0.0725
- accuracy: 0.9850 - val_loss: 0.4278 - val_accuracy: 0.8500
Epoch 10/20
25/25 [==============================] - 6s 230ms/step - loss: 0.0353
- accuracy: 0.9900 - val_loss: 0.6737 - val_accuracy: 0.8600
Epoch 11/20
25/25 [==============================] - 5s 206ms/step - loss: 0.0202
- accuracy: 0.9925 - val_loss: 0.7292 - val_accuracy: 0.8500
Epoch 12/20
25/25 [==============================] - 7s 263ms/step - loss: 0.0230
- accuracy: 0.9937 - val_loss: 0.5937 - val_accuracy: 0.8400
Epoch 13/20
25/25 [==============================] - 5s 214ms/step - loss: 0.0146
- accuracy: 0.9987 - val_loss: 0.8285 - val_accuracy: 0.8600

# Plotting the training and validation loss
plt.figure(figsize=(10, 5))

# Plotting training loss
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

# Plotting training and validation accuracy
plt.subplot(1, 2, 2)
```

```
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```




```
# Save the model
model.save('trained_model.h5')

# Save the model architecture
model_json = model.to_json()
with open('model_architecture.json', 'w') as json_file:
    json_file.write(model_json)

# Save the model weights
model.save_weights('model_weights.h5')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/
training.py:3103: UserWarning: You are saving your model as an HDF5
file via `model.save()`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')`.
  saving_api.save_model(
```