lsz8ukt6s

May 5, 2025

1 IMDB Sentiment Classification Using Keras

1.0.1 Step 1: Import Libraries

Explanation: We begin by importing all the libraries required for data processing, model building, and visualization.

[1]:
```
import pandas as pd
import re

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Dense, Dropout,␣
↪GlobalAveragePooling1D
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

import seaborn as sns
```

1.0.2 Step 2: Load Dataset

Explanation: We load the dataset `imdb_master.csv` using `pandas`. The file is encoded using `'ISO-8859-1'`.

[2]:
```
df = pd.read_csv("datasets/imdb_master.csv", encoding='ISO-8859-1')
df.head()
```

[2]:
```
   Unnamed: 0  type                                             review label  \
0           0  test  Once again Mr. Costner has dragged out a movie…   neg
1           1  test  This is an example of why the majority of acti…   neg
2           2  test  First of all I hate those moronic rappers, who…   neg
3           3  test  Not even the Beatles could write songs everyon…   neg
4           4  test  Brass pictures (movies is not a fitting word f…   neg

          file
0      0_2.txt
1  10000_4.txt
```

```
2  10001_1.txt
3  10002_3.txt
4  10003_3.txt
```

1.0.3 Step 3: Drop Unnecessary Columns

[3]:
```
df = df.drop(["Unnamed: 0", "file"], axis=1)
df.head()
```

[3]:

	type	review	label
0	test	Once again Mr. Costner has dragged out a movie…	neg
1	test	This is an example of why the majority of acti…	neg
2	test	First of all I hate those moronic rappers, who…	neg
3	test	Not even the Beatles could write songs everyon…	neg
4	test	Brass pictures (movies is not a fitting word f…	neg

1.0.4 Step 4: Explore Label Distribution (Before Filtering)

[4]:
```
df['label'].value_counts()
```

[4]:
```
label
unsup    50000
neg      25000
pos      25000
Name: count, dtype: int64
```

1.0.5 Step 5: Filter Dataset

[5]:
```
df = df.query("type == 'train' and label in ['pos', 'neg']")
df['label'].value_counts()
```

[5]:
```
label
neg    12500
pos    12500
Name: count, dtype: int64
```

1.0.6 Step 6: Separate Features and Labels

[6]:
```
texts = df['review']
y = df['label']
```

1.0.7 Step 7: Encode Labels

[7]:
```
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
```

1.0.8 Step 8: Tokenize the Text

[8]:
```
vocab_size = 10000
tokenizer = Tokenizer(num_words=vocab_size)
tokenizer.fit_on_texts(texts)
```

1.0.9 Step 9: Convert Text to Sequences

[9]:
```
X = tokenizer.texts_to_sequences(texts)
```

1.0.10 Step 10: Pad Sequences

[10]:
```
maxlen = 200
X = pad_sequences(X, maxlen=maxlen)
```

1.0.11 Step 11: Train-Test Split

[11]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

1.0.12 Step 12: Build the Model

[12]:
```
model = Sequential([
    Embedding(input_dim=10000, output_dim=64, input_length=200),
    GlobalAveragePooling1D(),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])
```

```
C:\Users\Acer\AppData\Roaming\Python\Python312\site-
packages\keras\src\layers\core\embedding.py:90: UserWarning: Argument
`input_length` is deprecated. Just remove it.
  warnings.warn(
```

1.0.13 Step 13: Compile the Model

[13]:
```
model.compile(optimizer='adam', loss='binary_crossentropy',␣
↪metrics=['accuracy'])
```

1.0.14 Step 14: Train the Model

[14]:
```
model.fit(X_train, y_train, epochs=5, batch_size=64, validation_data=(X_test,␣
↪y_test))
```

```
Epoch 1/5
313/313             4s 7ms/step -
accuracy: 0.6151 - loss: 0.6425 - val_accuracy: 0.8506 - val_loss: 0.3596
Epoch 2/5
313/313             2s 7ms/step -
accuracy: 0.8719 - loss: 0.3177 - val_accuracy: 0.8778 - val_loss: 0.2962
Epoch 3/5
313/313             2s 7ms/step -
accuracy: 0.9074 - loss: 0.2438 - val_accuracy: 0.8812 - val_loss: 0.2876
Epoch 4/5
313/313             3s 8ms/step -
accuracy: 0.9254 - loss: 0.2004 - val_accuracy: 0.8884 - val_loss: 0.2885
Epoch 5/5
313/313             2s 8ms/step -
accuracy: 0.9426 - loss: 0.1635 - val_accuracy: 0.8848 - val_loss: 0.3010
```

[14]:
```
<keras.src.callbacks.history.History at 0x2164f911b50>
```

1.0.15 Step 15: Evaluate Model Accuracy

[15]:
```
accuracy = model.evaluate(X_test, y_test)
```

```
157/157             0s 2ms/step -
accuracy: 0.8868 - loss: 0.2987
```

1.0.16 Step 16: Make Predictions

[16]:
```
predictions = (model.predict(X_test[:5]) > 0.5).astype(int)
```

```
1/1             0s 76ms/step
```

1.0.17 Step 17: Display Predictions vs Actual Labels

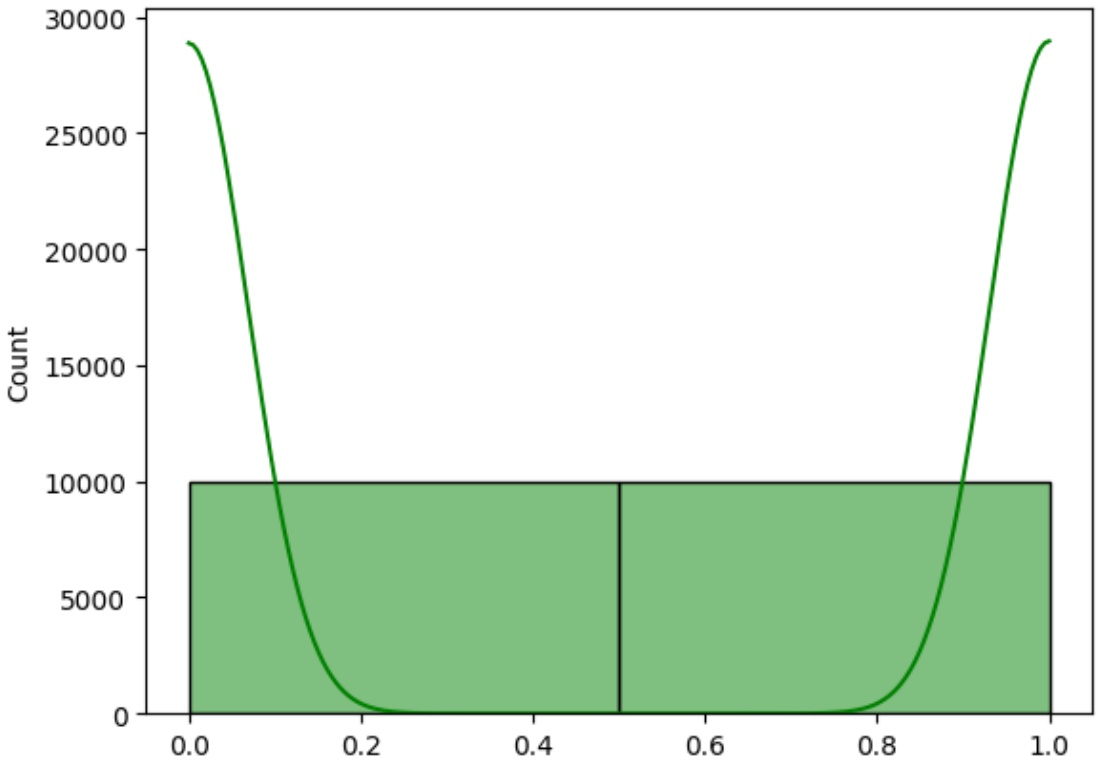
[17]:
```
for i in range(5):
    print(f"Predicted: {'Positive' if predictions[i][0] == 1 else 'Negative'} |␣
↪Actual: {'Positive' if y_test[i] == 1 else 'Negative'}")
```

```
Predicted: Negative | Actual: Negative
Predicted: Positive | Actual: Positive
Predicted: Positive | Actual: Negative
Predicted: Negative | Actual: Positive
Predicted: Positive | Actual: Positive
```

1.0.18 Step 18: Visualize Training Label Distribution

[18]:
```
sns.histplot(y_train, bins=2, kde=True, color='green')
```

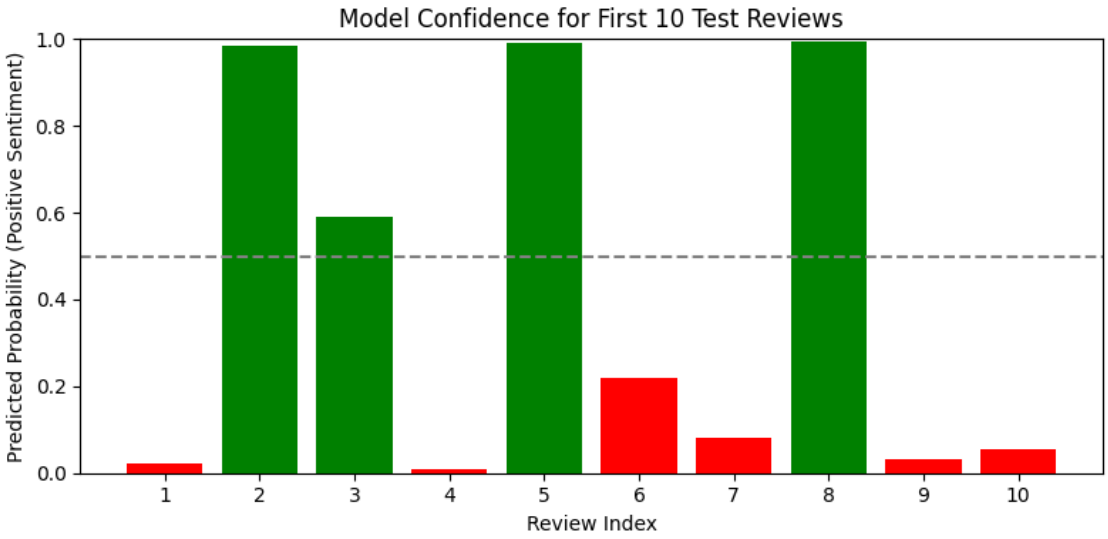
[18]:
```
<Axes: ylabel='Count'>
```



[20]:
```
# Get raw prediction probabilities
import matplotlib.pyplot as plt
probs = model.predict(X_test[:10])

# Plotting
plt.figure(figsize=(8, 4))
plt.bar(range(1, 11), probs.flatten(), color=['green' if p > 0.5 else 'red' for␣
 ↪p in probs])
plt.axhline(0.5, color='gray', linestyle='--')
plt.title("Model Confidence for First 10 Test Reviews")
plt.xlabel("Review Index")
plt.ylabel("Predicted Probability (Positive Sentiment)")
plt.xticks(range(1, 11))
plt.ylim(0, 1)
plt.tight_layout()
plt.show()
```

```
1/1                 0s 41ms/step
```

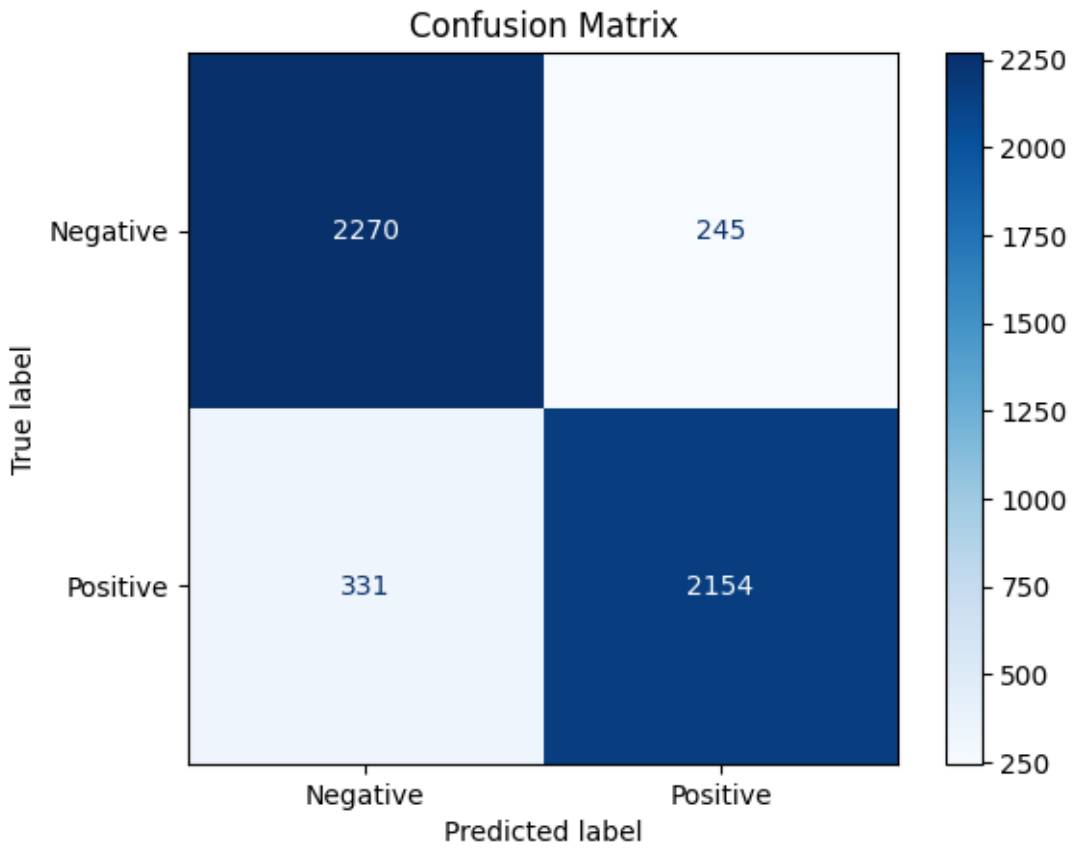


[21]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Get all predictions for test set
y_pred = (model.predict(X_test) > 0.5).astype(int)

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Negative",␣
  ↪"Positive"])
disp.plot(cmap='Blues')
plt.title("Confusion Matrix")
plt.show()
```

```
157/157             0s 1ms/step
```



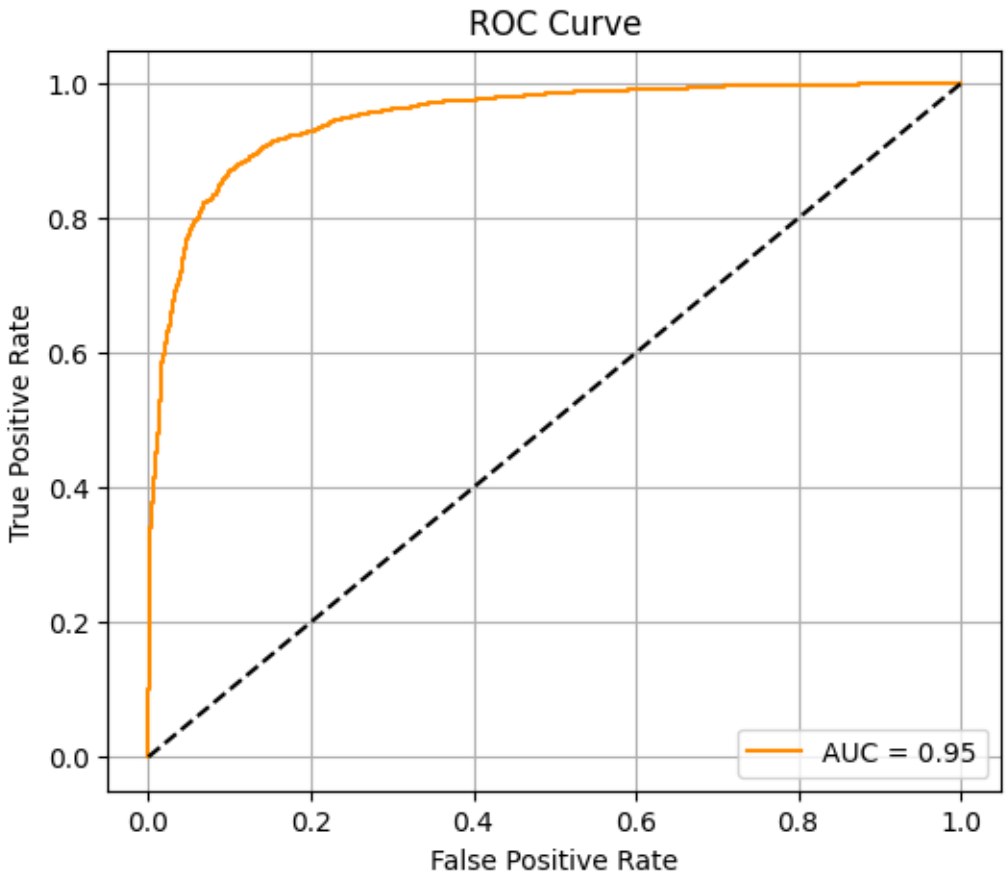
[22]:
```
from sklearn.metrics import roc_curve, auc

# Get probabilities
y_proba = model.predict(X_test).ravel()

# ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_proba)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, label=f'AUC = {roc_auc:.2f}', color='darkorange')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend(loc="lower right")
plt.grid(True)
plt.show()
```

```
157/157             0s 1ms/step
```


ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
AUC = 0.95