1. Define the Problem

Define the Problem

Input Data: The dataset consists of movie reviews, each labeled as positive or negative sentiment. This textual data serves as the input.

Type of Problem: The task is a binary classification problem focused on sentiment analysis. We aim to categorize each movie review into one of two sentiment classes: positive or negative.

Objective: The goal is to predict the sentiment of movie reviews accurately. By training a model on this dataset, we hope to understand and classify the emotions expressed in the text data effectively.

2. Choosing a Measure of Success

Choosing a Measure of Success

To evaluate our models, we will consider the following metrics:

- **Accuracy:** As a primary metric to measure the overall success rate of our predictions.
- **Precision and Recall:** These metrics will help us understand the quality of our positive predictions and the model's ability to capture the positive class respectively.
- **ROC AUC:** The Area Under the Receiver Operating Characteristic Curve (ROC AUC) will be used to evaluate the model's ability to discriminate between the classes.

.

3. Deciding on an Evaluation Protocol

Deciding on an Evaluation Protocol

We will employ a **hold-out validation set** approach for initial model evaluation. This method involves splitting the dataset into training and validation sets to train the model and assess its performance on unseen data.

Additionally, to ensure the robustness of our model evaluation, we will utilize **K-fold cross-validation**. This technique divides the data into K subsets, training the model K times, each time using a different subset as the validation set and the remaining data for training. .

4. Data Preparation

Preparing your Data

The preprocessing steps will include cleaning the text data, converting it into numerical format through tokenization, and then padding the sequences to a fixed length. This transformation is crucial for preparing the input data for our neural network models.

In [11]:

```
# Import necessary libraries
import pandas as pd
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import numpy as np
import re

# Function to clean the text data
def clean_text(text):
    """Clean text by removing non-alphabetic characters and lowercasing."""
    text = re.sub(r'[^a-zA-Z\s]', '', text, re.I|re.A)
    text = text.lower()
    text = text.strip()
    return text

# Load the dataset
df = pd.read_csv('movie_review.csv')

# Clean the text data
df['cleaned_text'] = df['text'].apply(clean_text)

# Tokenize and pad the cleaned text data
def tokenize_and_pad(texts, max_words=10000, max_len=100):
    tokenizer = Tokenizer(num_words=max_words, oov_token='<OOV>')
    tokenizer.fit_on_texts(texts)
    sequences = tokenizer.texts_to_sequences(texts)
    sequences_padded = pad_sequences(sequences, maxlen=max_len, padding='post')
    return sequences_padded, tokenizer

X, tokenizer = tokenize_and_pad(df['cleaned_text'].tolist())

# Convert labels to numerical format
y = df['tag'].map({'pos': 1, 'neg': 0}).values
```

Part 5: Developing a Model that Does Better than a Baseline

Developing a Model that Does Better than a Baseline

The objective here is to develop a neural network model that surpasses a common-sense baseline in terms of accuracy. The baseline model is a simple yet effective architecture designed to provide a reference point for improvement.

Common-Sense Baseline

Given the binary nature of our classification problem, a naive common-sense baseline could be the proportion of the most frequent class in the dataset. However, for a more challenging baseline, we aim for our models to learn from the text data's nuances, surpassing simple heuristics.

Baseline Model Architecture

- **Last Layer Activation:** Sigmoid, suitable for binary classification.
- **Loss Function:** Binary crossentropy, ideal for binary classification problems.
- **Optimizer:** Adam, with a default learning rate

In [27]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, GlobalAveragePooling1D, Dense, Dropo
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import roc_curve, auc, confusion_matrix, classification_report
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import re

# Define model creation functions
def create_baseline_model():
    model = Sequential([
        Embedding(input_dim=10000, output_dim=8, input_length=100),
        GlobalAveragePooling1D(),
        Dense(16, activation='relu'),
        Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'
    return model

def create_enhanced_model():
    model = Sequential([
        Embedding(input_dim=10000, output_dim=16, input_length=100),
        Dropout(0.5),
        GlobalAveragePooling1D(),
        Dropout(0.5),
        Dense(64, activation='relu'),
        Dropout(0.5),
        Dense(1, activation='sigmoid')
```

```
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'
    return model

# K-Fold Cross-Validation Setup
skf = StratifiedKFold(n_splits=5, shuffle=True)

for train_index, test_index in skf.split(X, y):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Baseline Model
    baseline_model = create_baseline_model()
```

```
Epoch 1/20
1618/1618 - 2s - loss: 0.6819 - accuracy: 0.5632 - val_loss: 0.6553 - val_accuracy:
0.6305 - 2s/epoch - 1ms/step
Epoch 2/20
1618/1618 - 2s - loss: 0.6130 - accuracy: 0.6711 - val_loss: 0.6045 - val_accuracy:
0.6741 - 2s/epoch - 1ms/step
Epoch 3/20
1618/1618 - 2s - loss: 0.5553 - accuracy: 0.7185 - val_loss: 0.5913 - val_accuracy:
0.6818 - 2s/epoch - 1ms/step
Epoch 4/20
1618/1618 - 2s - loss: 0.5245 - accuracy: 0.7348 - val_loss: 0.5921 - val_accuracy:
0.6831 - 2s/epoch - 1ms/step
Epoch 5/20
1618/1618 - 2s - loss: 0.5063 - accuracy: 0.7471 - val_loss: 0.5956 - val_accuracy:
0.6895 - 2s/epoch - 1ms/step
Epoch 6/20
1618/1618 - 2s - loss: 0.4931 - accuracy: 0.7545 - val_loss: 0.6061 - val_accuracy:
0.6870 - 2s/epoch - 1ms/step
Epoch 7/20
1618/1618 - 2s - loss: 0.4835 - accuracy: 0.7617 - val_loss: 0.6045 - val_accuracy:
0.6883 - 2s/epoch - 1ms/step
Epoch 8/20
1618/1618 - 2s - loss: 0.4764 - accuracy: 0.7661 - val_loss: 0.6149 - val_accuracy:
0.6888 - 2s/epoch - 1ms/step
Epoch 9/20
1618/1618 - 2s - loss: 0.4705 - accuracy: 0.7692 - val_loss: 0.6254 - val_accuracy:
0.6830 - 2s/epoch - 1ms/step
Epoch 10/20
1618/1618 - 2s - loss: 0.4666 - accuracy: 0.7697 - val_loss: 0.6245 - val_accuracy:
0.6883 - 2s/epoch - 1ms/step
Epoch 11/20
1618/1618 - 2s - loss: 0.4636 - accuracy: 0.7711 - val_loss: 0.6312 - val_accuracy:
0.6850 - 2s/epoch - 1ms/step
Epoch 12/20
1618/1618 - 2s - loss: 0.4598 - accuracy: 0.7751 - val_loss: 0.6446 - val_accuracy:
0.6765 - 2s/epoch - 1ms/step
Epoch 13/20
1618/1618 - 2s - loss: 0.4571 - accuracy: 0.7769 - val_loss: 0.6412 - val_accuracy:
0.6846 - 2s/epoch - 1ms/step
Epoch 14/20
1618/1618 - 2s - loss: 0.4546 - accuracy: 0.7770 - val_loss: 0.6452 - val_accuracy:
0.6836 - 2s/epoch - 1ms/step
Epoch 15/20
1618/1618 - 2s - loss: 0.4535 - accuracy: 0.7762 - val_loss: 0.6547 - val_accuracy:
0.6783 - 2s/epoch - 1ms/step
Epoch 16/20
1618/1618 - 2s - loss: 0.4520 - accuracy: 0.7793 - val_loss: 0.6530 - val_accuracy:
0.6835 - 2s/epoch - 1ms/step
Epoch 17/20
1618/1618 - 2s - loss: 0.4495 - accuracy: 0.7805 - val_loss: 0.6579 - val_accuracy:
0.6816 - 2s/epoch - 1ms/step
Epoch 18/20
1618/1618 - 2s - loss: 0.4492 - accuracy: 0.7790 - val_loss: 0.6626 - val_accuracy:
0.6817 - 2s/epoch - 1ms/step
Epoch 19/20
1618/1618 - 2s - loss: 0.4466 - accuracy: 0.7805 - val_loss: 0.6849 - val_accuracy:
```

```
0.6704 - 2s/epoch - 1ms/step
Epoch 20/20
1618/1618 - 2s - loss: 0.4466 - accuracy: 0.7819 - val_loss: 0.6713 - val_accuracy:
0.6811 - 2s/epoch - 1ms/step
Epoch 1/20
1618/1618 - 3s - loss: 0.6904 - accuracy: 0.5299 - val_loss: 0.6784 - val_accuracy:
0.5807 - 3s/epoch - 2ms/step
Epoch 2/20
1618/1618 - 3s - loss: 0.6521 - accuracy: 0.6170 - val_loss: 0.6214 - val_accuracy:
0.6585 - 3s/epoch - 2ms/step
Epoch 3/20
1618/1618 - 3s - loss: 0.6046 - accuracy: 0.6715 - val_loss: 0.6040 - val_accuracy:
0.6698 - 3s/epoch - 2ms/step
Epoch 4/20
1618/1618 - 3s - loss: 0.5783 - accuracy: 0.6971 - val_loss: 0.6039 - val_accuracy:
0.6656 - 3s/epoch - 2ms/step
Epoch 5/20
1618/1618 - 3s - loss: 0.5639 - accuracy: 0.7086 - val_loss: 0.5875 - val_accuracy:
0.6857 - 3s/epoch - 2ms/step
Epoch 6/20
1618/1618 - 3s - loss: 0.5537 - accuracy: 0.7153 - val_loss: 0.5887 - val_accuracy:
0.6877 - 3s/epoch - 2ms/step
Epoch 7/20
1618/1618 - 3s - loss: 0.5468 - accuracy: 0.7218 - val_loss: 0.5868 - val_accuracy:
0.6839 - 3s/epoch - 2ms/step
Epoch 8/20
1618/1618 - 3s - loss: 0.5414 - accuracy: 0.7262 - val_loss: 0.5852 - val_accuracy:
0.6887 - 3s/epoch - 2ms/step
Epoch 9/20
1618/1618 - 3s - loss: 0.5337 - accuracy: 0.7308 - val_loss: 0.5854 - val_accuracy:
0.6917 - 3s/epoch - 2ms/step
Epoch 10/20
1618/1618 - 3s - loss: 0.5291 - accuracy: 0.7362 - val_loss: 0.5858 - val_accuracy:
0.6844 - 3s/epoch - 2ms/step
Epoch 11/20
1618/1618 - 3s - loss: 0.5266 - accuracy: 0.7367 - val_loss: 0.5876 - val_accuracy:
0.6924 - 3s/epoch - 2ms/step
Epoch 12/20
1618/1618 - 3s - loss: 0.5219 - accuracy: 0.7390 - val_loss: 0.5878 - val_accuracy:
0.6915 - 3s/epoch - 2ms/step
Epoch 13/20
1618/1618 - 3s - loss: 0.5181 - accuracy: 0.7404 - val_loss: 0.5884 - val_accuracy:
0.6906 - 3s/epoch - 2ms/step
Epoch 14/20
1618/1618 - 3s - loss: 0.5191 - accuracy: 0.7425 - val_loss: 0.5897 - val_accuracy:
0.6916 - 3s/epoch - 2ms/step
Epoch 15/20
1618/1618 - 3s - loss: 0.5157 - accuracy: 0.7421 - val_loss: 0.5894 - val_accuracy:
0.6880 - 3s/epoch - 2ms/step
Epoch 16/20
1618/1618 - 3s - loss: 0.5106 - accuracy: 0.7447 - val_loss: 0.5941 - val_accuracy:
0.6900 - 3s/epoch - 2ms/step
Epoch 17/20
1618/1618 - 3s - loss: 0.5104 - accuracy: 0.7473 - val_loss: 0.6016 - val_accuracy:
0.6826 - 3s/epoch - 2ms/step
Epoch 18/20
```

```
1618/1618 - 3s - loss: 0.5077 - accuracy: 0.7496 - val_loss: 0.5988 - val_accuracy:
0.6907 - 3s/epoch - 2ms/step
Epoch 19/20
1618/1618 - 3s - loss: 0.5101 - accuracy: 0.7472 - val_loss: 0.5921 - val_accuracy:
0.6900 - 3s/epoch - 2ms/step
Epoch 20/20
1618/1618 - 3s - loss: 0.5061 - accuracy: 0.7513 - val_loss: 0.5948 - val_accuracy:
0.6904 - 3s/epoch - 2ms/step
Epoch 1/20
1618/1618 - 2s - loss: 0.6851 - accuracy: 0.5580 - val_loss: 0.6727 - val_accuracy:
0.5772 - 2s/epoch - 1ms/step
Epoch 2/20
1618/1618 - 2s - loss: 0.6217 - accuracy: 0.6629 - val_loss: 0.6113 - val_accuracy:
0.6697 - 2s/epoch - 1ms/step
Epoch 3/20
1618/1618 - 2s - loss: 0.5621 - accuracy: 0.7120 - val_loss: 0.5947 - val_accuracy:
0.6798 - 2s/epoch - 1ms/step
Epoch 4/20
1618/1618 - 2s - loss: 0.5287 - accuracy: 0.7338 - val_loss: 0.5930 - val_accuracy:
0.6833 - 2s/epoch - 1ms/step
Epoch 5/20
1618/1618 - 2s - loss: 0.5091 - accuracy: 0.7466 - val_loss: 0.6002 - val_accuracy:
0.6804 - 2s/epoch - 1ms/step
Epoch 6/20
1618/1618 - 2s - loss: 0.4951 - accuracy: 0.7547 - val_loss: 0.6024 - val_accuracy:
0.6850 - 2s/epoch - 1ms/step
Epoch 7/20
1618/1618 - 2s - loss: 0.4849 - accuracy: 0.7598 - val_loss: 0.6392 - val_accuracy:
0.6645 - 2s/epoch - 1ms/step
Epoch 8/20
1618/1618 - 2s - loss: 0.4770 - accuracy: 0.7651 - val_loss: 0.6157 - val_accuracy:
0.6849 - 2s/epoch - 1ms/step
Epoch 9/20
1618/1618 - 2s - loss: 0.4709 - accuracy: 0.7690 - val_loss: 0.6264 - val_accuracy:
0.6807 - 2s/epoch - 1ms/step
Epoch 10/20
1618/1618 - 2s - loss: 0.4664 - accuracy: 0.7706 - val_loss: 0.6358 - val_accuracy:
0.6761 - 2s/epoch - 1ms/step
Epoch 11/20
1618/1618 - 2s - loss: 0.4623 - accuracy: 0.7728 - val_loss: 0.6466 - val_accuracy:
0.6710 - 2s/epoch - 1ms/step
Epoch 12/20
1618/1618 - 2s - loss: 0.4600 - accuracy: 0.7746 - val_loss: 0.6421 - val_accuracy:
0.6799 - 2s/epoch - 1ms/step
Epoch 13/20
1618/1618 - 2s - loss: 0.4556 - accuracy: 0.7765 - val_loss: 0.6489 - val_accuracy:
0.6782 - 2s/epoch - 1ms/step
Epoch 14/20
1618/1618 - 2s - loss: 0.4539 - accuracy: 0.7777 - val_loss: 0.6523 - val_accuracy:
0.6765 - 2s/epoch - 1ms/step
Epoch 15/20
1618/1618 - 2s - loss: 0.4522 - accuracy: 0.7792 - val_loss: 0.6560 - val_accuracy:
0.6781 - 2s/epoch - 1ms/step
Epoch 16/20
1618/1618 - 2s - loss: 0.4498 - accuracy: 0.7812 - val_loss: 0.6642 - val_accuracy:
0.6754 - 2s/epoch - 1ms/step
```

```
Epoch 17/20
1618/1618 - 2s - loss: 0.4490 - accuracy: 0.7806 - val_loss: 0.6658 - val_accuracy:
0.6751 - 2s/epoch - 1ms/step
Epoch 18/20
1618/1618 - 2s - loss: 0.4467 - accuracy: 0.7810 - val_loss: 0.6721 - val_accuracy:
0.6742 - 2s/epoch - 1ms/step
Epoch 19/20
1618/1618 - 2s - loss: 0.4461 - accuracy: 0.7811 - val_loss: 0.6875 - val_accuracy:
0.6662 - 2s/epoch - 1ms/step
Epoch 20/20
1618/1618 - 2s - loss: 0.4443 - accuracy: 0.7830 - val_loss: 0.6877 - val_accuracy:
0.6647 - 2s/epoch - 1ms/step
Epoch 1/20
1618/1618 - 3s - loss: 0.6897 - accuracy: 0.5322 - val_loss: 0.6777 - val_accuracy:
0.5861 - 3s/epoch - 2ms/step
Epoch 2/20
1618/1618 - 3s - loss: 0.6545 - accuracy: 0.6148 - val_loss: 0.6284 - val_accuracy:
0.6506 - 3s/epoch - 2ms/step
Epoch 3/20
1618/1618 - 3s - loss: 0.6079 - accuracy: 0.6700 - val_loss: 0.6054 - val_accuracy:
0.6728 - 3s/epoch - 2ms/step
Epoch 4/20
1618/1618 - 3s - loss: 0.5818 - accuracy: 0.6940 - val_loss: 0.5956 - val_accuracy:
0.6826 - 3s/epoch - 2ms/step
Epoch 5/20
1618/1618 - 3s - loss: 0.5672 - accuracy: 0.7073 - val_loss: 0.5916 - val_accuracy:
0.6853 - 3s/epoch - 2ms/step
Epoch 6/20
1618/1618 - 3s - loss: 0.5557 - accuracy: 0.7151 - val_loss: 0.5906 - val_accuracy:
0.6858 - 3s/epoch - 2ms/step
Epoch 7/20
1618/1618 - 3s - loss: 0.5470 - accuracy: 0.7222 - val_loss: 0.5960 - val_accuracy:
0.6758 - 3s/epoch - 2ms/step
Epoch 8/20
1618/1618 - 3s - loss: 0.5392 - accuracy: 0.7293 - val_loss: 0.5869 - val_accuracy:
0.6853 - 3s/epoch - 2ms/step
Epoch 9/20
1618/1618 - 3s - loss: 0.5335 - accuracy: 0.7323 - val_loss: 0.5908 - val_accuracy:
0.6842 - 3s/epoch - 2ms/step
Epoch 10/20
1618/1618 - 3s - loss: 0.5297 - accuracy: 0.7349 - val_loss: 0.5933 - val_accuracy:
0.6829 - 3s/epoch - 2ms/step
Epoch 11/20
1618/1618 - 3s - loss: 0.5259 - accuracy: 0.7362 - val_loss: 0.5902 - val_accuracy:
0.6851 - 3s/epoch - 2ms/step
Epoch 12/20
1618/1618 - 3s - loss: 0.5234 - accuracy: 0.7377 - val_loss: 0.5917 - val_accuracy:
0.6862 - 3s/epoch - 2ms/step
Epoch 13/20
1618/1618 - 3s - loss: 0.5203 - accuracy: 0.7422 - val_loss: 0.5884 - val_accuracy:
0.6880 - 3s/epoch - 2ms/step
Epoch 14/20
1618/1618 - 3s - loss: 0.5173 - accuracy: 0.7419 - val_loss: 0.5954 - val_accuracy:
0.6839 - 3s/epoch - 2ms/step
Epoch 15/20
1618/1618 - 3s - loss: 0.5143 - accuracy: 0.7450 - val_loss: 0.6000 - val_accuracy:
```

```
0.6824 - 3s/epoch - 2ms/step
Epoch 16/20
1618/1618 - 3s - loss: 0.5136 - accuracy: 0.7450 - val_loss: 0.5922 - val_accuracy:
0.6870 - 3s/epoch - 2ms/step
Epoch 17/20
1618/1618 - 3s - loss: 0.5110 - accuracy: 0.7484 - val_loss: 0.6122 - val_accuracy:
0.6741 - 3s/epoch - 2ms/step
Epoch 18/20
1618/1618 - 3s - loss: 0.5072 - accuracy: 0.7500 - val_loss: 0.5985 - val_accuracy:
0.6860 - 3s/epoch - 2ms/step
Epoch 19/20
1618/1618 - 3s - loss: 0.5095 - accuracy: 0.7460 - val_loss: 0.6014 - val_accuracy:
0.6829 - 3s/epoch - 2ms/step
Epoch 20/20
1618/1618 - 3s - loss: 0.5056 - accuracy: 0.7502 - val_loss: 0.6021 - val_accuracy:
0.6832 - 3s/epoch - 2ms/step
Epoch 1/20
1618/1618 - 2s - loss: 0.6817 - accuracy: 0.5644 - val_loss: 0.6575 - val_accuracy:
0.6161 - 2s/epoch - 2ms/step
Epoch 2/20
1618/1618 - 2s - loss: 0.6073 - accuracy: 0.6742 - val_loss: 0.6038 - val_accuracy:
0.6751 - 2s/epoch - 1ms/step
Epoch 3/20
1618/1618 - 2s - loss: 0.5491 - accuracy: 0.7194 - val_loss: 0.6013 - val_accuracy:
0.6792 - 2s/epoch - 1ms/step
Epoch 4/20
1618/1618 - 2s - loss: 0.5179 - accuracy: 0.7414 - val_loss: 0.6002 - val_accuracy:
0.6812 - 2s/epoch - 1ms/step
Epoch 5/20
1618/1618 - 2s - loss: 0.4997 - accuracy: 0.7517 - val_loss: 0.6201 - val_accuracy:
0.6745 - 2s/epoch - 1ms/step
Epoch 6/20
1618/1618 - 2s - loss: 0.4876 - accuracy: 0.7590 - val_loss: 0.6110 - val_accuracy:
0.6830 - 2s/epoch - 1ms/step
Epoch 7/20
1618/1618 - 2s - loss: 0.4796 - accuracy: 0.7622 - val_loss: 0.6184 - val_accuracy:
0.6817 - 2s/epoch - 1ms/step
Epoch 8/20
1618/1618 - 2s - loss: 0.4721 - accuracy: 0.7672 - val_loss: 0.6242 - val_accuracy:
0.6817 - 2s/epoch - 1ms/step
Epoch 9/20
1618/1618 - 2s - loss: 0.4680 - accuracy: 0.7695 - val_loss: 0.6386 - val_accuracy:
0.6739 - 2s/epoch - 1ms/step
Epoch 10/20
1618/1618 - 2s - loss: 0.4634 - accuracy: 0.7727 - val_loss: 0.6366 - val_accuracy:
0.6782 - 2s/epoch - 1ms/step
Epoch 11/20
1618/1618 - 2s - loss: 0.4596 - accuracy: 0.7739 - val_loss: 0.6482 - val_accuracy:
0.6772 - 2s/epoch - 1ms/step
Epoch 12/20
1618/1618 - 2s - loss: 0.4573 - accuracy: 0.7744 - val_loss: 0.6453 - val_accuracy:
0.6776 - 2s/epoch - 1ms/step
Epoch 13/20
1618/1618 - 2s - loss: 0.4543 - accuracy: 0.7778 - val_loss: 0.6590 - val_accuracy:
0.6674 - 2s/epoch - 1ms/step
Epoch 14/20
```

```
1618/1618 - 2s - loss: 0.4535 - accuracy: 0.7773 - val_loss: 0.6571 - val_accuracy: 
0.6726 - 2s/epoch - 1ms/step
Epoch 15/20
1618/1618 - 2s - loss: 0.4505 - accuracy: 0.7781 - val_loss: 0.6674 - val_accuracy: 
0.6728 - 2s/epoch - 1ms/step
Epoch 16/20
1618/1618 - 2s - loss: 0.4498 - accuracy: 0.7784 - val_loss: 0.6646 - val_accuracy: 
0.6744 - 2s/epoch - 1ms/step
Epoch 17/20
1618/1618 - 2s - loss: 0.4475 - accuracy: 0.7805 - val_loss: 0.6934 - val_accuracy: 
0.6679 - 2s/epoch - 1ms/step
Epoch 18/20
1618/1618 - 2s - loss: 0.4471 - accuracy: 0.7789 - val_loss: 0.6729 - val_accuracy: 
0.6682 - 2s/epoch - 1ms/step
Epoch 19/20
1618/1618 - 2s - loss: 0.4459 - accuracy: 0.7809 - val_loss: 0.6765 - val_accuracy: 
0.6683 - 2s/epoch - 1ms/step
Epoch 20/20
1618/1618 - 2s - loss: 0.4446 - accuracy: 0.7817 - val_loss: 0.7029 - val_accuracy: 
0.6673 - 2s/epoch - 1ms/step
Epoch 1/20
1618/1618 - 3s - loss: 0.6897 - accuracy: 0.5271 - val_loss: 0.6761 - val_accuracy: 
0.5943 - 3s/epoch - 2ms/step
Epoch 2/20
1618/1618 - 3s - loss: 0.6518 - accuracy: 0.6184 - val_loss: 0.6261 - val_accuracy: 
0.6578 - 3s/epoch - 2ms/step
Epoch 3/20
1618/1618 - 3s - loss: 0.6062 - accuracy: 0.6710 - val_loss: 0.6072 - val_accuracy: 
0.6695 - 3s/epoch - 2ms/step
Epoch 4/20
1618/1618 - 3s - loss: 0.5811 - accuracy: 0.6942 - val_loss: 0.6014 - val_accuracy: 
0.6734 - 3s/epoch - 2ms/step
Epoch 5/20
1618/1618 - 3s - loss: 0.5657 - accuracy: 0.7081 - val_loss: 0.5973 - val_accuracy: 
0.6782 - 3s/epoch - 2ms/step
Epoch 6/20
1618/1618 - 3s - loss: 0.5531 - accuracy: 0.7175 - val_loss: 0.5904 - val_accuracy: 
0.6827 - 3s/epoch - 2ms/step
Epoch 7/20
1618/1618 - 3s - loss: 0.5433 - accuracy: 0.7263 - val_loss: 0.6002 - val_accuracy: 
0.6743 - 3s/epoch - 2ms/step
Epoch 8/20
1618/1618 - 3s - loss: 0.5368 - accuracy: 0.7276 - val_loss: 0.5916 - val_accuracy: 
0.6877 - 3s/epoch - 2ms/step
Epoch 9/20
1618/1618 - 3s - loss: 0.5327 - accuracy: 0.7329 - val_loss: 0.5951 - val_accuracy: 
0.6881 - 3s/epoch - 2ms/step
Epoch 10/20
1618/1618 - 3s - loss: 0.5301 - accuracy: 0.7365 - val_loss: 0.6121 - val_accuracy: 
0.6645 - 3s/epoch - 2ms/step
Epoch 11/20
1618/1618 - 3s - loss: 0.5258 - accuracy: 0.7374 - val_loss: 0.5966 - val_accuracy: 
0.6809 - 3s/epoch - 2ms/step
Epoch 12/20
1618/1618 - 3s - loss: 0.5204 - accuracy: 0.7407 - val_loss: 0.6001 - val_accuracy: 
0.6835 - 3s/epoch - 2ms/step
```

```
Epoch 13/20
1618/1618 - 3s - loss: 0.5183 - accuracy: 0.7433 - val_loss: 0.5986 - val_accuracy:
0.6822 - 3s/epoch - 2ms/step
Epoch 14/20
1618/1618 - 3s - loss: 0.5152 - accuracy: 0.7434 - val_loss: 0.6007 - val_accuracy:
0.6854 - 3s/epoch - 2ms/step
Epoch 15/20
1618/1618 - 3s - loss: 0.5127 - accuracy: 0.7462 - val_loss: 0.6141 - val_accuracy:
0.6721 - 3s/epoch - 2ms/step
Epoch 16/20
1618/1618 - 3s - loss: 0.5132 - accuracy: 0.7477 - val_loss: 0.6028 - val_accuracy:
0.6856 - 3s/epoch - 2ms/step
Epoch 17/20
1618/1618 - 3s - loss: 0.5112 - accuracy: 0.7460 - val_loss: 0.5987 - val_accuracy:
0.6818 - 3s/epoch - 2ms/step
Epoch 18/20
1618/1618 - 3s - loss: 0.5083 - accuracy: 0.7482 - val_loss: 0.6038 - val_accuracy:
0.6847 - 3s/epoch - 2ms/step
Epoch 19/20
1618/1618 - 3s - loss: 0.5071 - accuracy: 0.7494 - val_loss: 0.6056 - val_accuracy:
0.6795 - 3s/epoch - 2ms/step
Epoch 20/20
1618/1618 - 3s - loss: 0.5044 - accuracy: 0.7505 - val_loss: 0.6049 - val_accuracy:
0.6829 - 3s/epoch - 2ms/step
Epoch 1/20
1618/1618 - 2s - loss: 0.6759 - accuracy: 0.5725 - val_loss: 0.6439 - val_accuracy:
0.6404 - 2s/epoch - 1ms/step
Epoch 2/20
1618/1618 - 2s - loss: 0.5957 - accuracy: 0.6853 - val_loss: 0.5974 - val_accuracy:
0.6768 - 2s/epoch - 1ms/step
Epoch 3/20
1618/1618 - 2s - loss: 0.5404 - accuracy: 0.7248 - val_loss: 0.5922 - val_accuracy:
0.6833 - 2s/epoch - 1ms/step
Epoch 4/20
1618/1618 - 2s - loss: 0.5126 - accuracy: 0.7439 - val_loss: 0.5998 - val_accuracy:
0.6835 - 2s/epoch - 1ms/step
Epoch 5/20
1618/1618 - 2s - loss: 0.4961 - accuracy: 0.7525 - val_loss: 0.6084 - val_accuracy:
0.6826 - 2s/epoch - 1ms/step
Epoch 6/20
1618/1618 - 2s - loss: 0.4855 - accuracy: 0.7594 - val_loss: 0.6164 - val_accuracy:
0.6817 - 2s/epoch - 1ms/step
Epoch 7/20
1618/1618 - 2s - loss: 0.4758 - accuracy: 0.7656 - val_loss: 0.6198 - val_accuracy:
0.6818 - 2s/epoch - 1ms/step
Epoch 8/20
1618/1618 - 2s - loss: 0.4712 - accuracy: 0.7677 - val_loss: 0.6273 - val_accuracy:
0.6792 - 2s/epoch - 1ms/step
Epoch 9/20
1618/1618 - 2s - loss: 0.4653 - accuracy: 0.7711 - val_loss: 0.6380 - val_accuracy:
0.6755 - 2s/epoch - 1ms/step
Epoch 10/20
1618/1618 - 2s - loss: 0.4611 - accuracy: 0.7728 - val_loss: 0.6478 - val_accuracy:
0.6760 - 2s/epoch - 1ms/step
Epoch 11/20
1618/1618 - 2s - loss: 0.4594 - accuracy: 0.7746 - val_loss: 0.6470 - val_accuracy:
```

```
0.6788 - 2s/epoch - 1ms/step
Epoch 12/20
1618/1618 - 2s - loss: 0.4561 - accuracy: 0.7767 - val_loss: 0.6561 - val_accuracy:
0.6746 - 2s/epoch - 1ms/step
Epoch 13/20
1618/1618 - 2s - loss: 0.4534 - accuracy: 0.7767 - val_loss: 0.6576 - val_accuracy:
0.6786 - 2s/epoch - 1ms/step
Epoch 14/20
1618/1618 - 2s - loss: 0.4508 - accuracy: 0.7780 - val_loss: 0.6642 - val_accuracy:
0.6763 - 2s/epoch - 1ms/step
Epoch 15/20
1618/1618 - 2s - loss: 0.4495 - accuracy: 0.7791 - val_loss: 0.6785 - val_accuracy:
0.6717 - 2s/epoch - 1ms/step
Epoch 16/20
1618/1618 - 2s - loss: 0.4478 - accuracy: 0.7802 - val_loss: 0.6792 - val_accuracy:
0.6716 - 2s/epoch - 1ms/step
Epoch 17/20
1618/1618 - 2s - loss: 0.4472 - accuracy: 0.7813 - val_loss: 0.6743 - val_accuracy:
0.6765 - 2s/epoch - 1ms/step
Epoch 18/20
1618/1618 - 2s - loss: 0.4470 - accuracy: 0.7816 - val_loss: 0.6842 - val_accuracy:
0.6724 - 2s/epoch - 1ms/step
Epoch 19/20
1618/1618 - 2s - loss: 0.4453 - accuracy: 0.7828 - val_loss: 0.6804 - val_accuracy:
0.6745 - 2s/epoch - 1ms/step
Epoch 20/20
1618/1618 - 2s - loss: 0.4436 - accuracy: 0.7846 - val_loss: 0.6812 - val_accuracy:
0.6769 - 2s/epoch - 1ms/step
Epoch 1/20
1618/1618 - 3s - loss: 0.6898 - accuracy: 0.5310 - val_loss: 0.6779 - val_accuracy:
0.5818 - 3s/epoch - 2ms/step
Epoch 2/20
1618/1618 - 3s - loss: 0.6540 - accuracy: 0.6164 - val_loss: 0.6237 - val_accuracy:
0.6610 - 3s/epoch - 2ms/step
Epoch 3/20
1618/1618 - 3s - loss: 0.6046 - accuracy: 0.6732 - val_loss: 0.6047 - val_accuracy:
0.6718 - 3s/epoch - 2ms/step
Epoch 4/20
1618/1618 - 3s - loss: 0.5797 - accuracy: 0.6937 - val_loss: 0.5986 - val_accuracy:
0.6690 - 3s/epoch - 2ms/step
Epoch 5/20
1618/1618 - 3s - loss: 0.5617 - accuracy: 0.7098 - val_loss: 0.5902 - val_accuracy:
0.6850 - 3s/epoch - 2ms/step
Epoch 6/20
1618/1618 - 3s - loss: 0.5522 - accuracy: 0.7187 - val_loss: 0.5942 - val_accuracy:
0.6792 - 3s/epoch - 2ms/step
Epoch 7/20
1618/1618 - 3s - loss: 0.5456 - accuracy: 0.7220 - val_loss: 0.5918 - val_accuracy:
0.6825 - 3s/epoch - 2ms/step
Epoch 8/20
1618/1618 - 3s - loss: 0.5380 - accuracy: 0.7310 - val_loss: 0.5895 - val_accuracy:
0.6839 - 3s/epoch - 2ms/step
Epoch 9/20
1618/1618 - 3s - loss: 0.5327 - accuracy: 0.7326 - val_loss: 0.5935 - val_accuracy:
0.6843 - 3s/epoch - 2ms/step
Epoch 10/20
```

```
1618/1618 - 3s - loss: 0.5275 - accuracy: 0.7363 - val_loss: 0.5942 - val_accuracy:
0.6826 - 3s/epoch - 2ms/step
Epoch 11/20
1618/1618 - 3s - loss: 0.5222 - accuracy: 0.7388 - val_loss: 0.5930 - val_accuracy:
0.6867 - 3s/epoch - 2ms/step
Epoch 12/20
1618/1618 - 3s - loss: 0.5195 - accuracy: 0.7410 - val_loss: 0.5923 - val_accuracy:
0.6855 - 3s/epoch - 2ms/step
Epoch 13/20
1618/1618 - 3s - loss: 0.5162 - accuracy: 0.7415 - val_loss: 0.6114 - val_accuracy:
0.6761 - 3s/epoch - 2ms/step
Epoch 14/20
1618/1618 - 3s - loss: 0.5166 - accuracy: 0.7443 - val_loss: 0.5972 - val_accuracy:
0.6832 - 3s/epoch - 2ms/step
Epoch 15/20
1618/1618 - 3s - loss: 0.5130 - accuracy: 0.7453 - val_loss: 0.6151 - val_accuracy:
0.6730 - 3s/epoch - 2ms/step
Epoch 16/20
1618/1618 - 3s - loss: 0.5101 - accuracy: 0.7472 - val_loss: 0.5955 - val_accuracy:
0.6857 - 3s/epoch - 2ms/step
Epoch 17/20
1618/1618 - 3s - loss: 0.5084 - accuracy: 0.7468 - val_loss: 0.5999 - val_accuracy:
0.6853 - 3s/epoch - 2ms/step
Epoch 18/20
1618/1618 - 3s - loss: 0.5080 - accuracy: 0.7476 - val_loss: 0.5955 - val_accuracy:
0.6831 - 3s/epoch - 2ms/step
Epoch 19/20
1618/1618 - 3s - loss: 0.5065 - accuracy: 0.7511 - val_loss: 0.6028 - val_accuracy:
0.6826 - 3s/epoch - 2ms/step
Epoch 20/20
1618/1618 - 3s - loss: 0.5035 - accuracy: 0.7518 - val_loss: 0.6060 - val_accuracy:
0.6808 - 3s/epoch - 2ms/step
Epoch 1/20
1618/1618 - 3s - loss: 0.6799 - accuracy: 0.5660 - val_loss: 0.6518 - val_accuracy:
0.6279 - 3s/epoch - 2ms/step
Epoch 2/20
1618/1618 - 2s - loss: 0.6036 - accuracy: 0.6795 - val_loss: 0.6048 - val_accuracy:
0.6642 - 2s/epoch - 1ms/step
Epoch 3/20
1618/1618 - 2s - loss: 0.5477 - accuracy: 0.7191 - val_loss: 0.5899 - val_accuracy:
0.6804 - 2s/epoch - 1ms/step
Epoch 4/20
1618/1618 - 2s - loss: 0.5193 - accuracy: 0.7410 - val_loss: 0.5915 - val_accuracy:
0.6823 - 2s/epoch - 1ms/step
Epoch 5/20
1618/1618 - 2s - loss: 0.5010 - accuracy: 0.7518 - val_loss: 0.5982 - val_accuracy:
0.6845 - 2s/epoch - 1ms/step
Epoch 6/20
1618/1618 - 2s - loss: 0.4891 - accuracy: 0.7587 - val_loss: 0.6054 - val_accuracy:
0.6865 - 2s/epoch - 1ms/step
Epoch 7/20
1618/1618 - 2s - loss: 0.4808 - accuracy: 0.7625 - val_loss: 0.6131 - val_accuracy:
0.6842 - 2s/epoch - 1ms/step
Epoch 8/20
1618/1618 - 2s - loss: 0.4741 - accuracy: 0.7667 - val_loss: 0.6167 - val_accuracy:
0.6824 - 2s/epoch - 1ms/step
```

```
Epoch 9/20
1618/1618 - 2s - loss: 0.4705 - accuracy: 0.7676 - val_loss: 0.6220 - val_accuracy:
0.6839 - 2s/epoch - 1ms/step
Epoch 10/20
1618/1618 - 2s - loss: 0.4655 - accuracy: 0.7712 - val_loss: 0.6378 - val_accuracy:
0.6772 - 2s/epoch - 1ms/step
Epoch 11/20
1618/1618 - 2s - loss: 0.4624 - accuracy: 0.7734 - val_loss: 0.6343 - val_accuracy:
0.6833 - 2s/epoch - 1ms/step
Epoch 12/20
1618/1618 - 2s - loss: 0.4595 - accuracy: 0.7744 - val_loss: 0.6348 - val_accuracy:
0.6812 - 2s/epoch - 1ms/step
Epoch 13/20
1618/1618 - 2s - loss: 0.4565 - accuracy: 0.7779 - val_loss: 0.6410 - val_accuracy:
0.6829 - 2s/epoch - 1ms/step
Epoch 14/20
1618/1618 - 2s - loss: 0.4557 - accuracy: 0.7784 - val_loss: 0.6473 - val_accuracy:
0.6740 - 2s/epoch - 1ms/step
Epoch 15/20
1618/1618 - 2s - loss: 0.4534 - accuracy: 0.7801 - val_loss: 0.6460 - val_accuracy:
0.6809 - 2s/epoch - 1ms/step
Epoch 16/20
1618/1618 - 2s - loss: 0.4516 - accuracy: 0.7805 - val_loss: 0.6488 - val_accuracy:
0.6802 - 2s/epoch - 1ms/step
Epoch 17/20
1618/1618 - 2s - loss: 0.4509 - accuracy: 0.7796 - val_loss: 0.6520 - val_accuracy:
0.6809 - 2s/epoch - 1ms/step
Epoch 18/20
1618/1618 - 2s - loss: 0.4496 - accuracy: 0.7818 - val_loss: 0.6542 - val_accuracy:
0.6795 - 2s/epoch - 1ms/step
Epoch 19/20
1618/1618 - 2s - loss: 0.4487 - accuracy: 0.7824 - val_loss: 0.6596 - val_accuracy:
0.6749 - 2s/epoch - 1ms/step
Epoch 20/20
1618/1618 - 2s - loss: 0.4476 - accuracy: 0.7833 - val_loss: 0.6649 - val_accuracy:
0.6741 - 2s/epoch - 1ms/step
Epoch 1/20
1618/1618 - 3s - loss: 0.6898 - accuracy: 0.5300 - val_loss: 0.6764 - val_accuracy:
0.5980 - 3s/epoch - 2ms/step
Epoch 2/20
1618/1618 - 3s - loss: 0.6495 - accuracy: 0.6214 - val_loss: 0.6376 - val_accuracy:
0.6298 - 3s/epoch - 2ms/step
Epoch 3/20
1618/1618 - 3s - loss: 0.6047 - accuracy: 0.6759 - val_loss: 0.6017 - val_accuracy:
0.6751 - 3s/epoch - 2ms/step
Epoch 4/20
1618/1618 - 3s - loss: 0.5787 - accuracy: 0.6955 - val_loss: 0.5938 - val_accuracy:
0.6772 - 3s/epoch - 2ms/step
Epoch 5/20
1618/1618 - 3s - loss: 0.5637 - accuracy: 0.7076 - val_loss: 0.5893 - val_accuracy:
0.6833 - 3s/epoch - 2ms/step
Epoch 6/20
1618/1618 - 3s - loss: 0.5551 - accuracy: 0.7166 - val_loss: 0.5876 - val_accuracy:
0.6831 - 3s/epoch - 2ms/step
Epoch 7/20
1618/1618 - 3s - loss: 0.5470 - accuracy: 0.7236 - val_loss: 0.5886 - val_accuracy:
```

```
0.6859 - 3s/epoch - 2ms/step
Epoch 8/20
1618/1618 - 3s - loss: 0.5406 - accuracy: 0.7261 - val_loss: 0.5856 - val_accuracy:
0.6863 - 3s/epoch - 2ms/step
Epoch 9/20
1618/1618 - 3s - loss: 0.5342 - accuracy: 0.7333 - val_loss: 0.5860 - val_accuracy:
0.6868 - 3s/epoch - 2ms/step
Epoch 10/20
1618/1618 - 3s - loss: 0.5295 - accuracy: 0.7354 - val_loss: 0.5896 - val_accuracy:
0.6845 - 3s/epoch - 2ms/step
Epoch 11/20
1618/1618 - 3s - loss: 0.5253 - accuracy: 0.7396 - val_loss: 0.5902 - val_accuracy:
0.6836 - 3s/epoch - 2ms/step
Epoch 12/20
1618/1618 - 3s - loss: 0.5228 - accuracy: 0.7408 - val_loss: 0.5870 - val_accuracy:
0.6877 - 3s/epoch - 2ms/step
Epoch 13/20
1618/1618 - 3s - loss: 0.5214 - accuracy: 0.7415 - val_loss: 0.5895 - val_accuracy:
0.6887 - 3s/epoch - 2ms/step
Epoch 14/20
1618/1618 - 3s - loss: 0.5194 - accuracy: 0.7441 - val_loss: 0.5972 - val_accuracy:
0.6805 - 3s/epoch - 2ms/step
Epoch 15/20
1618/1618 - 3s - loss: 0.5136 - accuracy: 0.7444 - val_loss: 0.5913 - val_accuracy:
0.6857 - 3s/epoch - 2ms/step
Epoch 16/20
1618/1618 - 3s - loss: 0.5138 - accuracy: 0.7465 - val_loss: 0.5970 - val_accuracy:
0.6856 - 3s/epoch - 2ms/step
Epoch 17/20
1618/1618 - 3s - loss: 0.5118 - accuracy: 0.7462 - val_loss: 0.5927 - val_accuracy:
0.6858 - 3s/epoch - 2ms/step
Epoch 18/20
1618/1618 - 3s - loss: 0.5097 - accuracy: 0.7469 - val_loss: 0.5915 - val_accuracy:
0.6877 - 3s/epoch - 2ms/step
Epoch 19/20
1618/1618 - 3s - loss: 0.5084 - accuracy: 0.7483 - val_loss: 0.5938 - val_accuracy:
0.6852 - 3s/epoch - 2ms/step
Epoch 20/20
1618/1618 - 3s - loss: 0.5063 - accuracy: 0.7500 - val_loss: 0.5926 - val_accuracy:
0.6885 - 3s/epoch - 2ms/step
```

Part 6. Evaluating Models

In [30]:

```
def plot_accuracy_and_roc(history, y_true, y_pred, title_prefix):
    # Plot training & validation accuracy values
    plt.figure(figsize=(14, 6))

    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title(f'{title_prefix} Model Accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Validation'], loc='upper left')
```

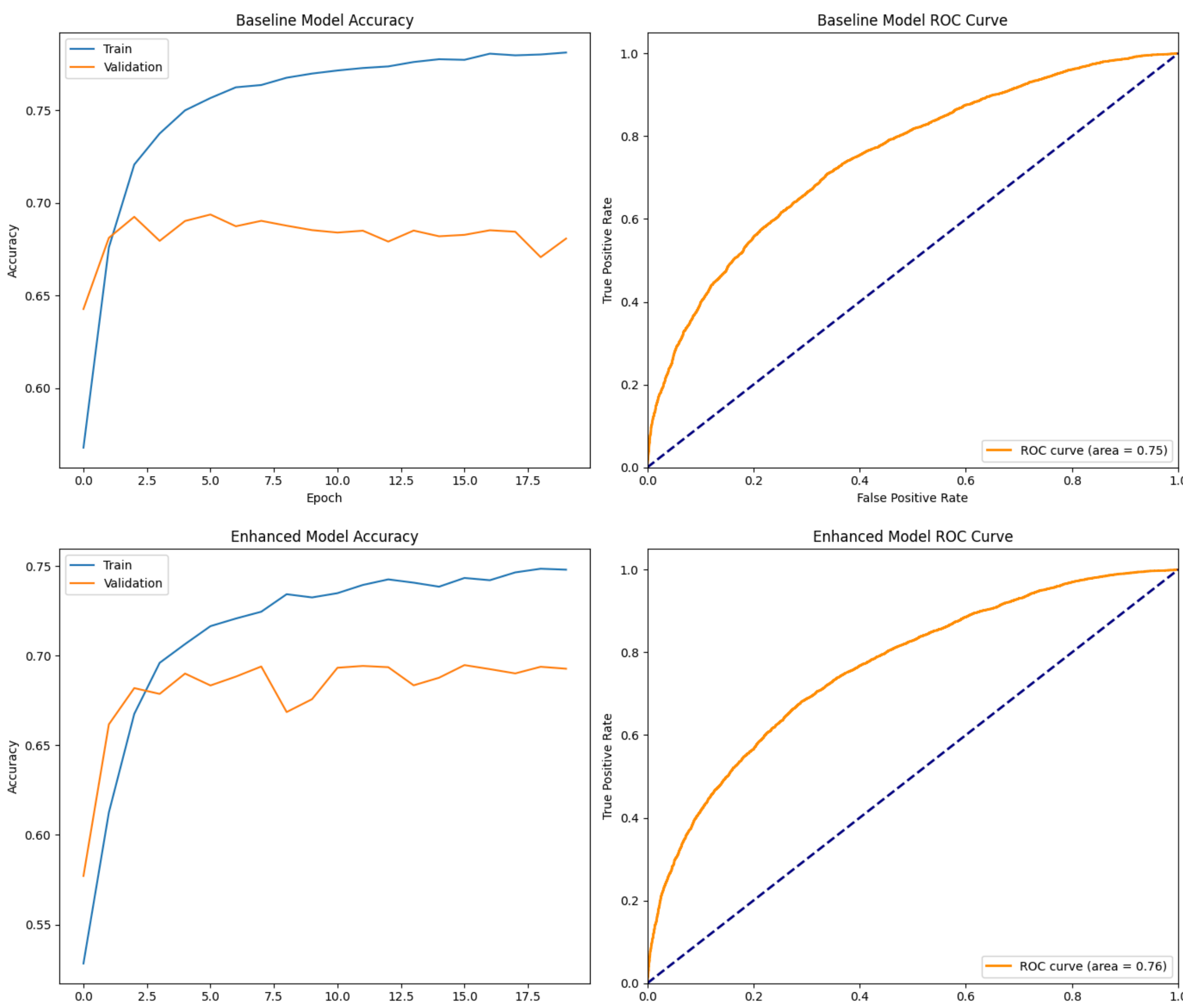
```python
    # Plot ROC curve
    fpr, tpr, thresholds = roc_curve(y_true, y_pred)
    roc_auc = auc(fpr, tpr)

    plt.subplot(1, 2, 2)
    plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(f'{title_prefix} Model ROC Curve')
    plt.legend(loc="lower right")

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

In [31]:

```python
# Ensure y_pred_baseline and y_pred_enhanced are the predicted probabilities for th
plot_accuracy_and_roc(history_baseline, y_val, y_pred_baseline, 'Baseline')
plot_accuracy_and_roc(history_enhanced, y_val, y_pred_enhanced, 'Enhanced')
```



Part 7. Testing the Best Model

In [39]:
```
def predict_sentiment(text, tokenizer, model):
    cleaned_text = clean_text(text)
    sequences = tokenizer.texts_to_sequences([cleaned_text])
    X_input = pad_sequences(sequences, maxlen=100, padding='post')
    prediction = model.predict(X_input)
    return "Positive Sentiment" if prediction[0] >= 0.5 else "Negative Sentiment"

# User input prediction (this part is interactive and meant to be run as part of a
user_input = input("Enter your text: ")
print(predict_sentiment(user_input, tokenizer, enhanced_model))
```

```
1/1 [==============================] - 0s 14ms/step
Positive Sentiment
```

Error Analysis

In [42]:
```
predictions = enhanced_model.predict(X_test)
predictions_binary = (predictions > 0.5).astype(int)

# Identify indices where predictions do not match actual labels
error_indices = np.where(predictions_binary.flatten() != y_test)[0]
```

```
405/405 [==============================] - 0s 756us/step
```

In [41]:
```
def display_error_analysis(error_indices, X_test, y_test, predictions):
    for index in error_indices[:10]:  # Display the first 10 errors for brevity
        print("Review Text: ", df['cleaned_text'][index])  # Assuming df is your Da
        print("Actual Sentiment: ", "Positive" if y_test[index] == 1 else "Negative
        print("Predicted Sentiment: ", "Positive" if predictions[index] > 0.5 else
        print("\n")

# Call the function to display the error analysis
display_error_analysis(error_indices, X_test, y_test, predictions)
```

Review Text: films adapted from comic books have had plenty of success whether the yre about superheroes batman superman spawn   or geared toward kids casper or t he arthouse crowd ghost world   but theres never really been a comic book like from hell before
Actual Sentiment: Positive
Predicted Sentiment: Negative

Review Text: to say moore and campbell thoroughly researched the subject of jack th e ripper would be like saying michael jackson is starting to look a little odd
Actual Sentiment: Positive
Predicted Sentiment: Negative

Review Text: the book or graphic novel   if you will is over pages long and inc ludes nearly more that consist of nothing but footnotes
Actual Sentiment: Positive
Predicted Sentiment: Negative

Review Text: in other words dont dismiss this film because of its source
Actual Sentiment: Positive
Predicted Sentiment: Negative

Review Text: the ghetto in question is of course whitechapel in londons east end
Actual Sentiment: Positive
Predicted Sentiment: Negative

Review Text: upon arriving in whitechapel he befriends an unfortunate named mary k elly heather graham say it isnt so and proceeds to investigate the horribly grues ome crimes that even the police surgeon cant stomach
Actual Sentiment: Positive
Predicted Sentiment: Negative

Review Text: in the comic they dont bother cloaking the identity of the ripper bu t screenwriters terry hayes vertical limit and rafael yglesias les mis rables d o a good job of keeping him hidden from viewers until the very end
Actual Sentiment: Positive
Predicted Sentiment: Negative

Review Text: its funny to watch the locals blindly point the finger of blame at jew s and indians because after all an englishman could never be capable of committing such ghastly acts
Actual Sentiment: Positive
Predicted Sentiment: Negative

Review Text: and from hells ending had me whistling the stonecutters song from the simpsons for days   who holds back the electric carwho made steve guttenberg a star
Actual Sentiment: Positive
Predicted Sentiment: Negative

Review Text: the print i saw wasnt completely finished both color and music had not been finalized so no comments about marilyn manson   but cinematographer peter deming dont say a word ably captures the dreariness of victorianera london and helped make the flashy killing scenes remind me of the crazy flashbacks in twin peaks even though the violence in the film pales in comparison to that in the blackandwhite comic
Actual Sentiment: Positive
Predicted Sentiment: Negative