**CODE 2 – IMDB DATASET**  
import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import LabelEncoder

df = pd.read\_csv("data\_2.csv")

X = df['review'].values

y = df['sentiment'].values

label\_encoder = LabelEncoder()

y = label\_encoder.fit\_transform(y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

from sklearn.feature\_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()

X\_train = vectorizer.fit\_transform(X\_train)

X\_test = vectorizer.transform(X\_test)

model = keras.Sequential([

layers.Dense(32, activation='relu', kernel\_regularizer=keras.regularizers.l2(0.01), input\_shape=(X\_train.shape[1],)),

layers.Dropout(0.3), # Dropout to reduce overfitting

layers.Dense(16, activation='relu', kernel\_regularizer=keras.regularizers.l2(0.01)),

layers.Dropout(0.3),

layers.Dense(1, activation='sigmoid') # Binary classification

])

model.compile(optimizer=keras.optimizers.Adam(learning\_rate=0.001), loss='binary\_crossentropy', metrics=['accuracy'])

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

history = model.fit(X\_train, y\_train, epochs=20, batch\_size=32, validation\_data=(X\_test, y\_test))

test\_loss, test\_acc = model.evaluate(X\_test, y\_test)

print(f'Test Accuracy: {test\_acc:.4f}')

y\_pred = (model.predict(X\_test) > 0.5).astype("int32")

print("Sample Predictions:", y\_pred[:5].flatten())

import matplotlib.pyplot as plt

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title('Model Accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

plt.subplot(1, 2, 2)

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.title('Model Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.show()