# Ex No: 6 BUILD A RECURRENT NEURAL NETWORK

**AIM:**

To build a recurrent neural network with Keras/TensorFlow.

# PROCEDURE:

1. Download and load the dataset.
2. Perform analysis and preprocessing of the dataset.
3. Build a simple neural network model using Keras/TensorFlow.
4. Compile and fit the model.
5. Perform prediction with the test dataset.
6. Calculate performance metrics.

# PROGRAM:

from tensorflow.keras.datasets import imdb

# Load the IMDb dataset

(train\_data, train\_labels), (test\_data, test\_labels) = imdb.load\_data(num\_words=10000)

from tensorflow.keras.preprocessing.sequence import pad\_sequences

# Pad the sequences to ensure all inputs have the same length train\_data = pad\_sequences(train\_data, maxlen=200) test\_data = pad\_sequences(test\_data, maxlen=200)

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, LSTM, Dense

# Build the RNN model model = Sequential()

model.add(Embedding(input\_dim=10000, output\_dim=32, input\_length=200)) model.add(LSTM(64, return\_sequences=False))

model.add(Dense(1, activation='sigmoid'))

model.summary()

model.compile(optimizer='adam',loss='binary\_crossentropy',metrics=['accuracy']) history = model.fit(train\_data, train\_labels,epochs=5,batch\_size=64,validation\_split=0.2)

test\_loss, test\_acc = model.evaluate(test\_data, test\_labels) print(f"Test accuracy: {test\_acc}")

predictions = model.predict(test\_data)

from sklearn.metrics import classification\_report, confusion\_matrix import matplotlib.pyplot as plt

# Classification report

y\_pred = (predictions > 0.5).astype("int32") print(classification\_report(test\_labels, y\_pred))

# Confusion matrix

cm = confusion\_matrix(test\_labels, y\_pred) print(cm)

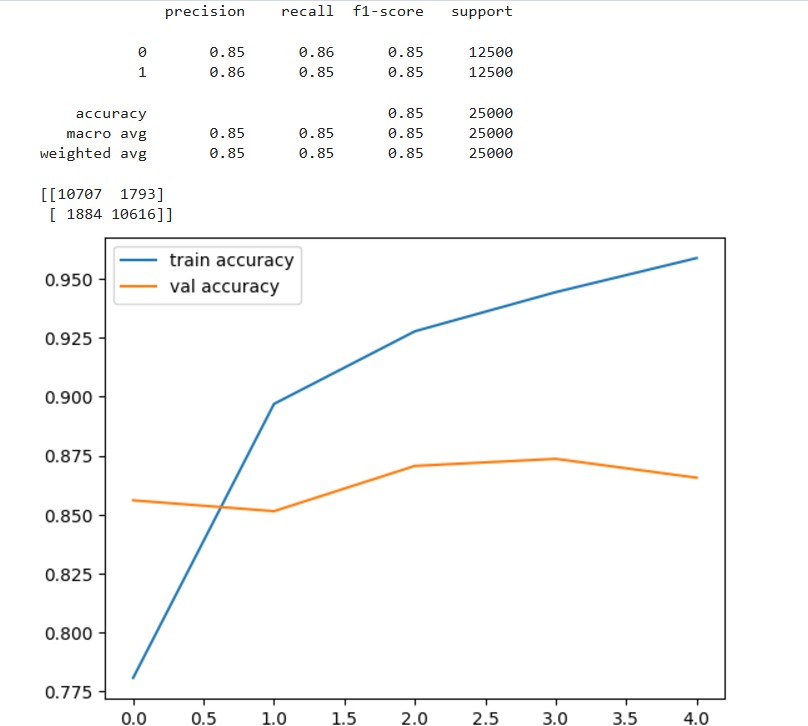
# Plotting accuracy and loss curves plt.plot(history.history['accuracy'], label='train accuracy') plt.plot(history.history['val\_accuracy'], label='val accuracy') plt.legend()

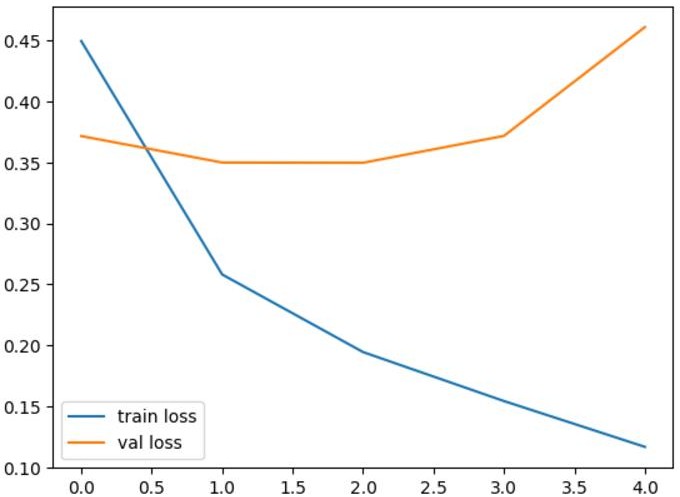
plt.show()

plt.plot(history.history['loss'], label='train loss') plt.plot(history.history['val\_loss'], label='val loss') plt.legend()

plt.show()

# OUTPUT:





**RESULT:**

Thus a recurrent neural network with Keras/TensorFlow is built.