Name : Muhammad Eri Mushthofa

PSU email: mmm7769@psu.edu

CLASSIFICATION TASK ON MOVIE DATASET

Dataset Source: https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset

1. Using RNN (Recurrent Neural Network)

This project involves the construction of a Recurrent Neural Network (RNN) using Keras, aimed at classifying text data. The data, sourced from 'train_data.txt,' is loaded into pandas DataFrame with columns like "ID," "Title," "Genre," and "Description." The pre-processing stage involves normalizing the text in the 'Description' column by converting it to lowercase and removing non-alphanumeric characters, a common practice in natural language processing. The 'Genre' column, the target variable, is transformed into numerical values using LabelEncoder.

To prepare the data for the RNN, a tokenizer is implemented, restricting its focus to the top 5000 words and converting the descriptions into sequences of integers. These sequences are then padded to a uniform length of 150 for consistency in training. The dataset is partially divided into training, validation, and test sets, with proportions of 60%, 20%, and 20%.

The RNN model architecture includes an Embedding layer, an LSTM (Long Short-Term Memory) layer, and a Dense layer. The LSTM and Dense layers' parameters, such as the number of units and the learning rate, are not preset but are subject to optimization through hyperparameter tuning using kerastuner. The tuning process, limited to 10 trials, seeks the best combination of LSTM units and learning rates based on validation accuracy.

The training employs early stopping, ceasing if the validation accuracy fails to improve for two consecutive epochs within a maximum of 5 epochs, thereby preventing overfitting. The optimal model is retrained and evaluated on the test set after hyperparameter tuning. The evaluation includes a classification report and an accuracy score, offering insights into the model's performance across different genres.

```
Reloading Tuner from my_dir/intro_to_kt/tuner0.json
Epoch 1/5
1017/1017 [==============================] - 233s 227ms/step - loss: 2.2146 - accuracy: 0.3542 - val_loss: 2.0253 - val_accuracy: 0.4370
Epoch 2/5
1017/1017 [==============================] - 215s 211ms/step - loss: 1.8622 - accuracy: 0.4729 - val_loss: 1.8250 - val_accuracy: 0.4854
Epoch 3/5
1017/1017 [==============================] - 212s 208ms/step - loss: 1.6400 - accuracy: 0.5289 - val_loss: 1.6813 - val_accuracy: 0.5178
Epoch 4/5
1017/1017 [==============================] - 210s 206ms/step - loss: 1.4915 - accuracy: 0.5687 - val_loss: 1.6402 - val_accuracy: 0.5242
Epoch 5/5
1017/1017 [==============================] - 211s 208ms/step - loss: 1.3725 - accuracy: 0.5998 - val_loss: 1.6199 - val_accuracy: 0.5330
339/339 [==============================] - 10s 29ms/step
```

	precision	recall	f1-score	support
0	0.27	0.11	0.16	266
1	0.30	0.06	0.10	97
2	0.31	0.03	0.06	144
3	0.12	0.02	0.03	118
4	0.00	0.00	0.00	64
5	0.44	0.62	0.51	1495
6	0.00	0.00	0.00	107
7	0.66	0.82	0.73	2607
8	0.56	0.66	0.60	2790
9	0.00	0.00	0.00	147
10	0.00	0.00	0.00	67
11	0.00	0.00	0.00	40
12	0.00	0.00	0.00	51
13	0.45	0.61	0.52	403
14	0.50	0.33	0.40	140
15	0.00	0.00	0.00	52
16	0.00	0.00	0.00	53
17	0.00	0.00	0.00	25
18	0.13	0.12	0.12	176
19	0.00	0.00	0.00	139
20	0.33	0.12	0.17	139
21	0.33	0.28	0.30	1011
22	0.51	0.19	0.27	97
23	0.00	0.00	0.00	85
24	0.36	0.12	0.18	326
25	0.00	0.00	0.00	24
26	0.69	0.67	0.68	180
accuracy			0.53	10843
macro avg	0.22	0.18	0.18	10843
weighted avg	0.47	0.53	0.48	10843

2. Using CNN (Convolutional Neural Network)

The project begins with loading data from 'train_data.txt' into a pandas DataFrame, structured with columns "ID," "Title," "Genre," and "Description." In preprocessing, the text in the 'Description' column undergoes standard normalization - conversion to lowercase and removal of non-alphanumeric characters. Meanwhile, the 'Genre' column is numerically encoded using LabelEncoder.

The next step involves tokenization, where a tokenizer is developed to convert the text descriptions into integer sequences, focusing on a vocabulary of the top 5000 words. These sequences are then padded to a standardized length of 150. The dataset is strategically divided into training, validation, and test sets, with a distribution of 60%, 20%, and 20%, respectively.

A CNN architecture is defined through a function for model building, incorporating hyperparameters like embedding dimension, filters, kernel size, and learning rate. The model's structure includes an Embedding layer, a Conv1D layer, a GlobalMaxPooling1D layer, and a Dense output layer. The hyperparameter tuning employs a Keras Tuner RandomSearch tuner, aiming for the highest validation accuracy. It experiments with up to 10 different configurations.

During the tuning phase, early stopping is utilized, halting training if there is no improvement in validation accuracy for two epochs. Following this, the best-performing model is identified, trained on the training data, and further validated over 5 epochs.

In the final stage, this model is assessed on the test set, with the primary metric being test accuracy. This comprehensive approach, involving careful preprocessing, hyperparameter tuning, and systematic evaluation, aims to optimize CNN's performance in classifying text data.

```
Trial 10 Complete [00h 04m 59s]
val_accuracy: 0.4372406303882599

Best val_accuracy So Far: 0.5417320132255554
Total elapsed time: 01h 12m 29s
Epoch 1/5
1017/1017 [==============================] - 184s 180ms/step - loss: 0.8948 - accuracy: 0.7449 - val_loss: 1.6976 - val_accuracy: 0.5368
Epoch 2/5
1017/1017 [==============================] - 173s 170ms/step - loss: 0.5812 - accuracy: 0.8467 - val_loss: 1.8509 - val_accuracy: 0.5344
Epoch 3/5
1017/1017 [==============================] - 173s 170ms/step - loss: 0.3175 - accuracy: 0.9297 - val_loss: 2.0542 - val_accuracy: 0.5204
Epoch 4/5
1017/1017 [==============================] - 185s 182ms/step - loss: 0.1439 - accuracy: 0.9794 - val_loss: 2.3171 - val_accuracy: 0.5201
Epoch 5/5
1017/1017 [==============================] - 170s 167ms/step - loss: 0.0570 - accuracy: 0.9966 - val_loss: 2.5582 - val_accuracy: 0.5168
339/339 [==============================] - 3s 8ms/step - loss: 2.6070 - accuracy: 0.5107
Test Accuracy: 0.5106520056724548
```

3. Conclusion

Some problems I encountered was the length of the running time that is why I set up a maximum trial of 10 and it still took at least an hour to finish. The best accuracy I got for RNN is 53% and the best accuracy I got for CNN is 54%