In [2]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/book-rating-prediction01/sample_submission.csv
/kaggle/input/book-rating-prediction01/Train.csv
/kaggle/input/book-rating-prediction01/Test.csv
```

1. Loading the Dataset

In [3]:
```
dataset = pd.read_csv("/kaggle/input/book-rating-prediction01/Train.csv",encoding="latin")
df = pd.DataFrame(dataset)
df.head()
```

Out[3]:

	user_id	book_id	review_id	review_text	date_added	date_updated	read_at	started_at	n_votes	n_comments	rating
0	e8730ce3ed5e9762038d160e23d47d79	2187	1c6af3913167e3d9d26be5a29d8df1aa	4.5 stars \n I loved this book! It was incredi...	Fri May 23 13:53:31 -0700 2014	Sun Apr 03 03:56:27 -0700 2016	Sun Mar 29 00:00:00 -0700 2015	Mon Aug 25 00:00:00 -0700 2014	0	0	4
1	26317e667d9141ad245dd2c18be52d77	47212	fa1f648451baf909683ea40bda3a06fe	I've had my dad and friend after me to read th...	Thu Jan 05 21:23:23 -0800 2017	Sun Jan 08 21:09:19 -0800 2017	Sun Jan 08 00:00:00 -0800 2017	Thu Jan 05 00:00:00 -0800 2017	1	0	4
2	bf9640b81a047ee70b4f918082492f1d	40483	ae2c43f1062ab1dae0ad51f5ea3c7c56	Anyone desiring to start an exploration of S&M...	Tue Feb 18 03:29:28 -0800 2014	Fri Feb 28 01:24:54 -0800 2014	Fri Feb 28 01:54:07 -0800 2014	Tue Feb 18 00:00:00 -0800 2014	1	1	3
3	34aa99d428ad98679c3e45d117243f55	19095025	3eb21a560e2afb02ace9e44d6fe76c8b	3.75 stars \n Mal is the best. He's crazy fun ...	Fri May 08 17:06:08 -0700 2015	Sun May 21 17:12:59 -0700 2017	Mon Sep 07 00:00:00 -0700 2015	Sun Sep 06 00:00:00 -0700 2015	1	0	4
4	6b3f929609c9d97628807d13b59b0b22	9464746	6caffed66bddb57550e777f04823fdd6	I would like to begin by saying how much I app...	Tue Feb 08 15:47:53 -0800 2011	Sat Nov 05 21:12:58 -0700 2011	Sat Nov 05 00:00:00 -0700 2011	Sun Oct 30 00:00:00 -0700 2011	0	0	5

2. PreProcessing

Data Preprocessing for NLP Model: In preparing the text data for our NLP model, following preprocessing functions are performed:

- Lowercasing: The text has been converted to lowercase to ensure uniformity and eliminate case sensitivity.
- HTML Tags and URLs Removal: HTML tags and URLs have been removed from the text to focus on meaningful content.
- Punctuation Cleaning: Punctuation marks have been removed from the text strings to streamline analysis and avoid interference with the model.
- Stopword Removal: Common stopwords, such as "and," "the," etc., have been eliminated to reduce noise and enhance the importance of significant words.

- Tokenization and Stemming: The text has been tokenized into individual words or tokens, and a stemming process has been applied to reduce words to their root form. This aids in standardization and consolidates related words.

In [4]:
```
new_df = df
new_df.head()
```

Out[4]:

	user_id	book_id	review_id	review_text	date_added	date_updated	read_at	started_at	n_votes	n_comments	rating
0	e8730ce3ed5e9762038d160e23d47d79	2187	1c6af3913167e3d9d26be5a29d8df1aa	4.5 stars \n I loved this book! It was incredi...	Fri May 23 13:53:31 -0700 2014	Sun Apr 03 03:56:27 -0700 2016	Sun Mar 29 00:00:00 -0700 2015	Mon Aug 25 00:00:00 -0700 2014	0	0	4
1	26317e667d9141ad245dd2c18be52d77	47212	fa1f648451baf909683ea40bda3a06fe	I've had my dad and friend after me to read th...	Thu Jan 05 21:23:23 -0800 2017	Sun Jan 08 21:09:19 -0800 2017	Sun Jan 08 00:00:00 -0800 2017	Thu Jan 05 00:00:00 -0800 2017	1	0	4
2	bf9640b81a047ee70b4f918082492f1d	40483	ae2c43f1062ab1dae0ad51f5ea3c7c56	Anyone desiring to start an exploration of S&M...	Tue Feb 18 03:29:28 -0800 2014	Fri Feb 28 01:24:54 -0800 2014	Fri Feb 28 01:54:07 -0800 2014	Tue Feb 18 00:00:00 -0800 2014	1	1	3
3	34aa99d428ad98679c3e45d117243f55	19095025	3eb21a560e2afb02ace9e44d6fe76c8b	3.75 stars \n Mal is the best. He's crazy fun ...	Fri May 08 17:06:08 -0700 2015	Sun May 21 17:12:59 -0700 2017	Mon Sep 07 00:00:00 -0700 2015	Sun Sep 06 00:00:00 -0700 2015	1	0	4
4	6b3f929609c9d97628807d13b59b0b22	9464746	6caffed66bddb57550e777f04823fdd6	I would like to begin by saying how much I app...	Tue Feb 08 15:47:53 -0800 2011	Sat Nov 05 21:12:58 -0700 2011	Sat Nov 05 00:00:00 -0700 2011	Sun Oct 30 00:00:00 -0700 2011	0	0	5

HTML Tag and urls removal

- In this function we have used the regular expressions to remove html tags, urls, tabs and new line characters.

In [5]:
```
import re

def remove_html_tags_and_urls(text):
    clean = re.sub(r'[\t\n]', '', text)
    clean = re.sub(r'<.*?>', '', clean)
    clean = re.sub(r'https?://\S+|www\.\S+', '', clean)
    return clean
```

Remove Punctuation

- We have remove the punctuation except the points in the integers to ensure that there is no loss of information

In [6]:
```
import string
string.punctuation
```

Out[6]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

In [7]:
```
st = '''!"#$%&\'()*+,-/:;<=>?@[\\]^_`{|}~'''
```

In [8]:
```
def remove_punc(text):
    return text.translate(str.maketrans('','', st))
```

Clean String

In [9]:
```
def clean_string(input_str):
    cleaned_str = ''.join(char for char in input_str if char.isalnum() or char.isspace() or char == '.')
    return cleaned_str
```

Except Dots

In [10]:
```
import re

def remove_dots_except_decimals(input_str):
    result = re.sub(r'(?<!\d)\.(?!\d)', '', input_str)
    result = re.sub(r'\.(?!\d)', '', result)
    return result
```

Stopwords Removal

- Here we have used the NLTK library's stopwords to remove the stopwords from the reviews to focus on the significant words more.

In [11]:
```
from nltk.corpus import stopwords

stop_words = set(stopwords.words('english'))

def remove_stopword(text):
    words = text.split()
    filtered_words = [word for word in words if word not in stop_words]
    return ' '.join(filtered_words)
```

Tokenization and Stemming

- Two-way stemming condenses words to their roots and retains potential reversibility, enhancing linguistic context preservation, aiding interpretability, and ensuring readability in the processed text.

In [13]:
```
from nltk.stem import PorterStemmer
class TwoWayStemmer:
    def __init__(self):
        self.stemmer = PorterStemmer()
        self.stem_dict = {}

    def stem_text(self, text):
        stemmed_text = []
        for word in text.split():
            stemmed_word = self.stemmer.stem(word)
            if stemmed_word not in self.stem_dict:
                self.stem_dict[stemmed_word] = [word]
            else:
                if word not in self.stem_dict[stemmed_word]:
                    self.stem_dict[stemmed_word].append(word)
            stemmed_text.append(stemmed_word)
        return ' '.join(stemmed_text)
```

Applying Functions

In [14]:
```
new_df['review_text'] = new_df['review_text'].str.lower()
new_df['review_text'].head()
```

Out[14]:
```
0    4.5 stars \n i loved this book! it was incredi...
1    i've had my dad and friend after me to read th...
2    anyone desiring to start an exploration of s&m...
3    3.75 stars \n mal is the best. he's crazy fun ...
4    i would like to begin by saying how much i app...
Name: review_text, dtype: object
```

In [15]:
```
new_df['review_text'] = new_df['review_text'].apply(remove_html_tags_and_urls)
new_df['review_text'].head()
```

Out[15]:
```
0    4.5 stars  i loved this book! it was incredibl...
1    i've had my dad and friend after me to read th...
2    anyone desiring to start an exploration of s&m...
3    3.75 stars  mal is the best. he's crazy fun an...
4    i would like to begin by saying how much i app...
Name: review_text, dtype: object
```

In [16]:
```
new_df['review_text'] = new_df['review_text'].apply(clean_string)
new_df['review_text'].head()
```

Out[16]:
```
0    4.5 stars  i loved this book it was incredibly...
1    ive had my dad and friend after me to read thi...
2    anyone desiring to start an exploration of sm ...
3    3.75 stars  mal is the best. hes crazy fun and...
4    i would like to begin by saying how much i app...
Name: review_text, dtype: object
```

In [17]:
```
new_df['review_text'] = new_df['review_text'].apply(remove_dots_except_decimals)
new_df['review_text'].head()
```

Out[17]:
```
0    4.5 stars  i loved this book it was incredibly...
1    ive had my dad and friend after me to read thi...
2    anyone desiring to start an exploration of sm ...
3    3.75 stars  mal is the best hes crazy fun and ...
4    i would like to begin by saying how much i app...
Name: review_text, dtype: object
```

In [18]:
```
new_df['review_text'] = new_df['review_text'].apply(remove_stopword)
new_df['review_text'].head()
```

Out[18]:
```
0    4.5 stars loved book incredibly intricate well...
1    ive dad friend read series years friend gave g...
2    anyone desiring start exploration sm erotica s...
3    3.75 stars mal best hes crazy fun says things ...
4    would like begin saying much appreciate amount...
Name: review_text, dtype: object
```

In [19]:
```
two_way_stemmer = TwoWayStemmer()

# Assuming 'train_df' is your training dataframe with 'review_text' column
new_df['review_text'] = new_df['review_text'].apply(lambda x: two_way_stemmer.stem_text(x))
```

MODEL TRAINING

- It imports necessary modules for text and categorical feature processing, including TfidfVectorizer for text, OneHotEncoder for categorical features.
- It sets up a ColumnTransformer to preprocess text data with TfidfVectorizer (5000 features, word and bi-gram range) and one-hot encodes a user ID feature.

- The neural network architecture isn't defined here but can concatenate processed inputs using Keras layers for later model building.
- train_test_split might be used later for data splitting. Overall, it's setting up a pipeline for feature preprocessing for a neural network model.

In [32]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
import pandas as pd

X_train, X_test, y_train, y_test = train_test_split(
    new_df[['user_id','review_text']], new_df['rating'], test_size=0.2, random_state=42
)
```

In [33]:
```
text_features = 'review_text'
user_id_feature = 'user_id'
votes_feature = 'n_votes'
comments_feature = 'n_comments'
```

In [50]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from keras.layers import Input, Dense, Concatenate
from keras.models import Model
import numpy as np


# Preprocessing
preprocessor = ColumnTransformer(
    transformers=[
        ('text', TfidfVectorizer(max_features=5000, ngram_range=(1, 2)), text_features),
        ('user_id', OneHotEncoder(handle_unknown='ignore'), [user_id_feature])
    ]
)


# Fit and transform the data using the preprocessor
X_processed = preprocessor.fit_transform(X_train)
```

In [51]:
```
# Split the data
X_train_tfidf, X_val_tfidf, y_train_tfidf, y_val = train_test_split(
    X_processed, y_train, test_size=0.1, random_state=42
)
```

Creating Custom Neural Network

- The provided code sets up a neural network using the Keras library. It constructs a Sequential model comprising dense layers, gradually reducing neurons. Leaky Rectified Linear Unit (LeakyReLU) activation functions introduce non-linearity in each layer. The output layer has six neurons for multi-class classification, utilizing softmax activation to output class probabilities. Additionally, it compiles the model, configuring it for training with the Adam optimizer (learning rate 0.001), categorical cross-entropy loss for multiclass problems, and accuracy as the evaluation metric. Overall, it builds a neural network architecture, ready for training on data with defined input dimensions and classification into multiple classes.

In [68]:
```
from keras.models import Sequential
from keras.layers import Dense, LeakyReLU
from keras.optimizers import Adam
```

```
# Create a Sequential model
model = Sequential()

# Add layers to the Sequential model

model.add(Dense(512, input_shape=(X_train_tfidf.shape[1],)))
model.add(LeakyReLU(alpha=0.2))

model.add(Dense(256))
model.add(LeakyReLU(alpha=0.2))

model.add(Dense(128))
model.add(LeakyReLU(alpha=0.2))

model.add(Dense(64))
model.add(LeakyReLU(alpha=0.2))

model.add(Dense(6, activation='softmax'))

# Compile the model
custom_optimizer = Adam(learning_rate=0.001)
model.compile(optimizer=custom_optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
```

In [56]:
```
from keras.utils import to_categorical

# Assuming y_train contains class indices
y_train_encoded = to_categorical(y_train_tfidf)
y_val_encoded = to_categorical(y_val)
```

In [ ]:
```
# Train the model
history = model.fit(
    X_train_tfidf, y_train_encoded,
    epochs=2, batch_size=32,
    validation_data=(X_val_tfidf, y_val_encoded)
)
```

```
Epoch 1/2
28350/28350 [==============================] - 408s 14ms/step - loss: 1.0499 - accuracy: 0.5517 - val_loss: 1.0016 - val_accuracy: 0.5744
Epoch 2/2
 6971/28350 [======>.......................] - ETA: 3:17 - loss: 0.9325 - accuracy: 0.6062
```

In [58]:
```
X_test_processed = preprocessor.transform(X_test)
y_test_encoded = to_categorical(y_test)
```

In [59]:
```
test_loss, test_accuracy = model.evaluate(X_test_processed, y_test_encoded)

print(f"Test Accuracy: {test_accuracy * 100:.2f}%")
print(f"Test Loss: {test_loss}")
```

```
3938/3938 [==============================] - 14s 4ms/step - loss: 1.0697 - accuracy: 0.5685
Test Accuracy: 56.85%
Test Loss: 1.0697451829910278
```

Predicting TEST DATA

- For prediction the test data has to be preprocessed as the training data was done. So applying the same preprocessing steps on the testing data and transforming it using the same column transform for prediction

In [60]:
```
test_df = pd.read_csv("/kaggle/input/book-rating-prediction01/Test.csv")
```

In [61]:
```
# Applying functions
test_df['review_text'] = test_df['review_text'].str.lower()
test_df['review_text'] = test_df['review_text'].apply(remove_html_tags_and_urls)
test_df['review_text'] = test_df['review_text'].apply(clean_string)
test_df['review_text'] = test_df['review_text'].apply(remove_dots_except_decimals)
test_df['review_text'] = test_df['review_text'].apply(remove_stopword)
```

In [62]:
```
# Assuming 'train_df' is your training dataframe with 'review_text' column
test_df['review_text'] = test_df['review_text'].apply(lambda x: two_way_stemmer.stem_text(x))
```

In [66]:
```
test_processed = preprocessor.transform(test_df[['user_id','review_text']])
```

In [ ]:
```
predictions = model.predict(test_processed)
predicted_classes = np.argmax(predictions, axis=1)
predicted_classes
```

In [65]:
```
# Assuming test_df['review_id'] and predictions are two-dimensional arrays of shape (270000, 1)
predictions_df = pd.DataFrame({'review_id': test_df['review_id'], 'rating': predicted_classes})

# Save the DataFrame to a CSV file
predictions_df.to_csv('predictions_final.csv', index=False)
```

In [ ]:
```
predictions_df['rating'].value_counts()
```

In [ ]: