AUTHOR STYLE CLASSIFICATION (LSTM)

Darius Dylan Govender

Table of Contents

Introduction

For this project, the aim was to develop a bot that was able to predict the author of a given text snippet, based purely on the writing style of the author. This analysis explores how Natural Language Processing (NLP) and machine learning particularly, LSTM (Long Short-Term Memory) can be applied to author classification. It is important to understand that a Long Short-Term Memory (LSTM) is a more advanced form of Hochreiter and Schmidhuber's Recurrent Neural Network (RNN) (GeeksForGeeks, 2025).

The project follows a pipeline similar to other NLP studies: data cleaning and preparation, text preprocessing, exploratory data analysis (EDA), model training, and evaluation. Using a dataset of texts labelled by author, I prepared the data for classification, trained with a LSTM model, and tuned hyperparameters to optimise performance. Ultimately, this model will be used in a console-based prediction bot that allows users to input any passage of text, and the model outputs the most likely author.

Data Overview and Cleaning

Data cleaning involves finding and removing missing, duplicate and/or irrelevant data (GeekForGeeks, 2025). Data cleaning aims to maintain data accuracy, consistence and is noise free (GeekForGeeks, 2025). This step is critical for ensuring data quality as well as well performing model.

First, all necessary libraries were installed and imported. This analysis made use of Spark therefore, in the first step Spark was also initialised. The second step was to perform an initial inspection of the data using multiple checks such as df.show() and df.printShcema() this showed the dataset consisted of 25671 rows and 4 columns. No missing values however there were duplicate values present within the dataset. These duplicate values were removed to reduce bias. Before moving on to encoding some issues were found with the author spelling (many different variations of the author names), thus this data was normalised. Next a class imbalance was identified and thus, 3 authors were dropped as they were too little samples of them present in the dataset. After the class imbalance was addressed, the distribution of the class sat at almost balance with 'JK Rolling' at 50.9% and 'Stephen King' at 49.1%, just a 1.8% difference. Finally, the authors were encoded using StringIndexer (Big Data Landscape, 2025)

Step	Action Performed	Outcome
Import and install	Import and install and necessary libraries	Data analysis environment setup
Load dataset	Spark was initialised and the dataset was loaded using Spark	Data analysis environment setup
Inspect	Ran df.show() and df.printSchema	25671 rows and 4 columns
Null values	Built a function to check for null values as this would have to be used multiple times in this analysis	No missing values
Duplicate values	Built a function to check for duplicate values as this would have to be used multiple times in this analysis	528 duplicates found, dropped all to reduce bias
Fixed Encoding issues	Fixed author name spelling and uniformed it with a switch statement	Uniformed ladled author name, prepared for encoding
Class imbalance	Authors with limited samples were removed as they didn’t even make up 10 percent.	Fixed class imbalance,
Feature encoding	Encoded author label with StringIndexer	All target labels encoded

Text preprocessing

Text preprocessing is an important stage in Natural Language Processing (NLP) which involves then cleaning and transforming raw text data into a format suitable for analysis and machine learning models. (Abdullah, 2024).

The following procedure was followed to ensure all text was pre-processed to standards:

1. The Spark data frame was converted to a Pandas data frame.
2. Basic text cleaning • In this step noise such as HTML tags, URLs, special characters, extra spaces are reduced so that the data can be standardised (Abdullah, 2024). Additionally, all contractions are expanded to standardise the data, and all text is converted to lowercase (Abdullah, 2024). All of this prevents a shewed analysis due to irrelevant symbols or formatting issues (Abdullah, 2024).
3. Lemmatisation • In this step lemmatisation is made use of to convert words to their base forms. This is done to treat variants of a word as a single feature (Abdullah, 2024). So as an example, the word running would become run. This helps prevent overfitting as well as enhance generalisation (Abdullah, 2024).

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is an important step in the data analysis process, in this process the aim is to understand patterns and find relationships by using statistical and visual tools (GeekForGeeks, 2025).

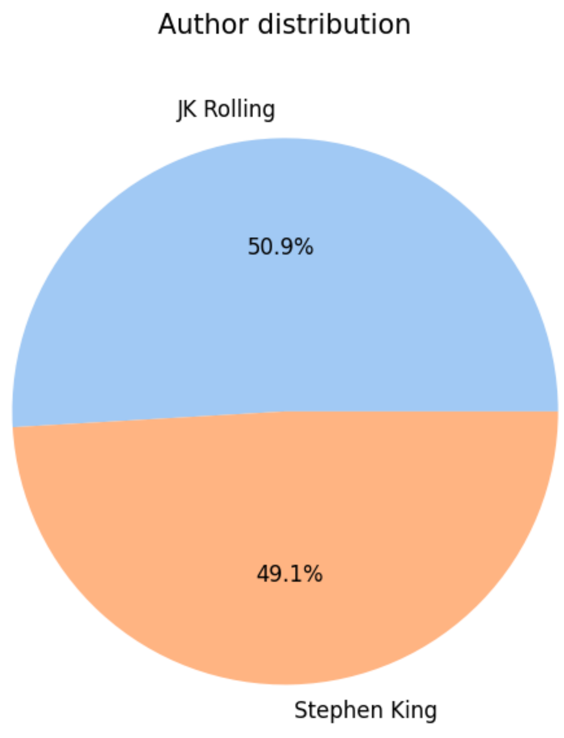
In this section it was broken down into the Univariate analysis and Bivariate analysis

1. Univariate analysis focuses on individual variables (GeekForGeeks, 2025).
2. Bivariate analysis focuses on feature relationships (GeekForGeeks, 2025).

Univariate analysis

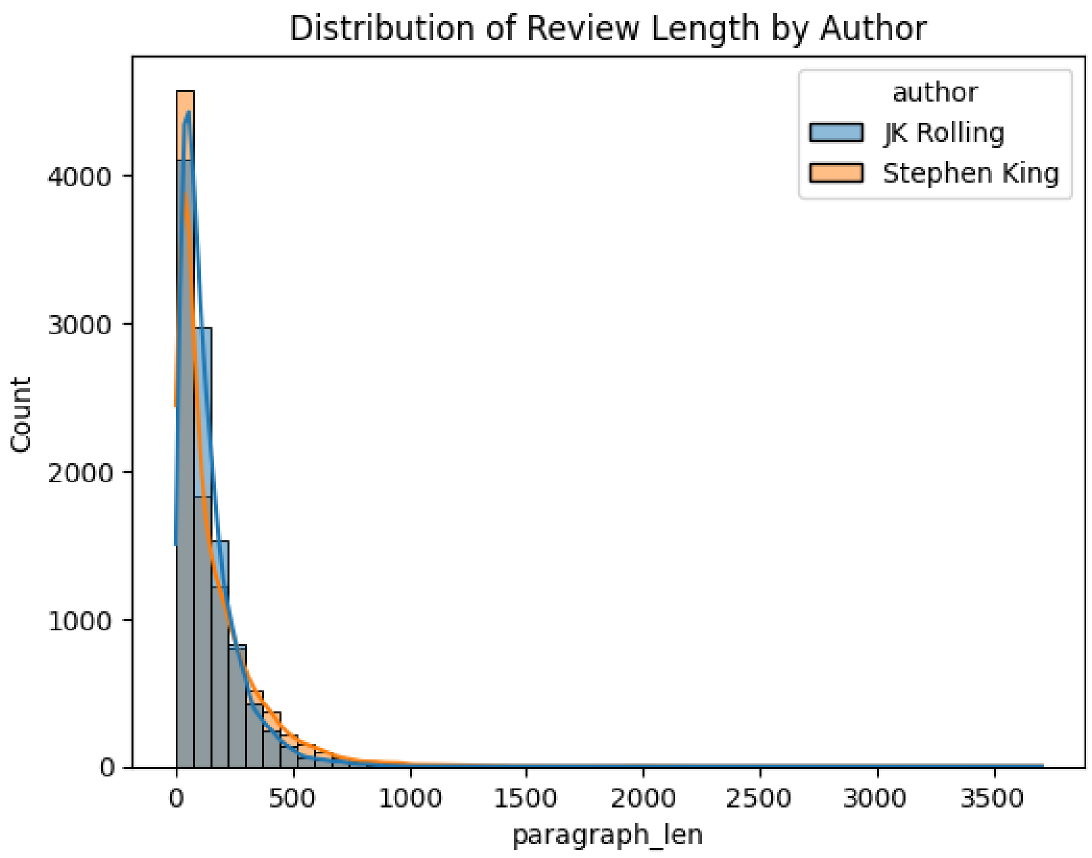
The univariate analysis looked to understand the distribution of features as well as factors such as review length and word frequency. In section pie charts, histograms and word clouds were used to visualise this data.

Class Distribution Analysis



This is a visual representation of the class imbalance, as displayed above the dataset is almost balance with 'JK Rolling' at 50.9% and 'Stephen King' at 49.1%, just a 1.8% difference.

Text Length Analysis

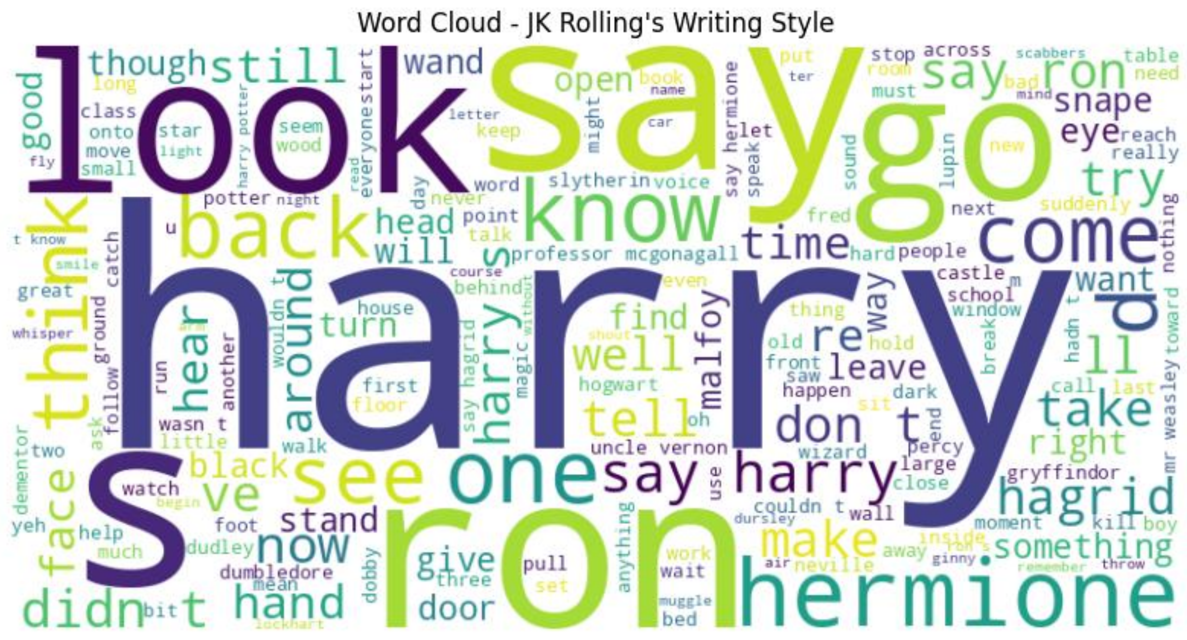


Based on the graph above which longs to understand the text length of reviews, it has found that:

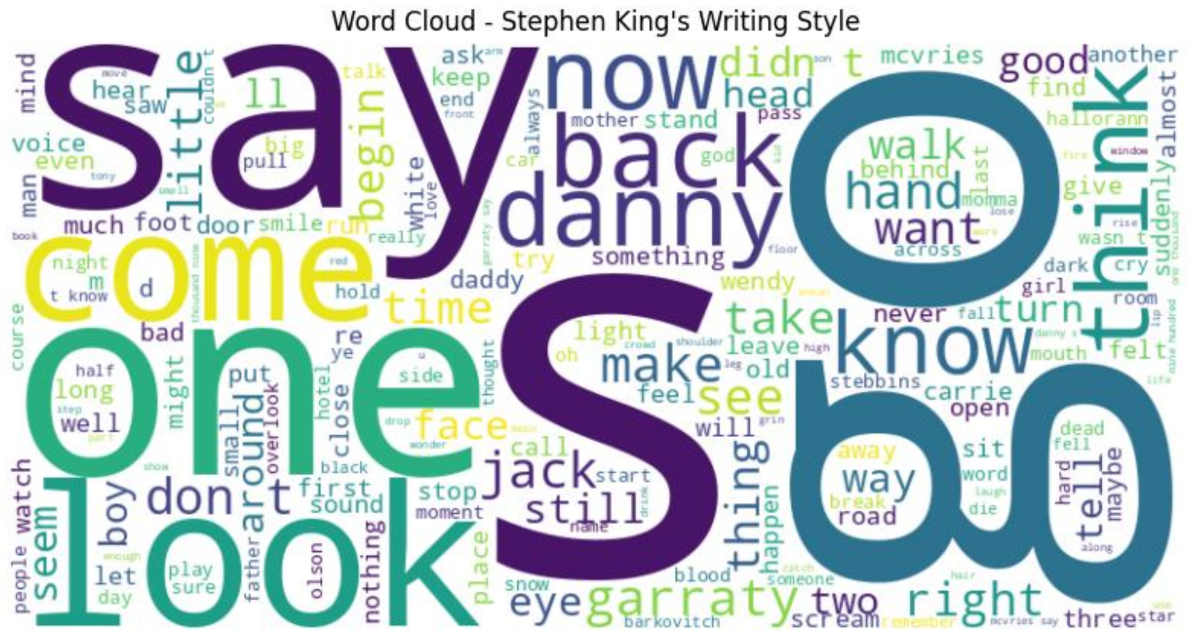
- Most of the reviews are quite short, with a median length of only 93 words.
- However, there is a high maximum value of 3,708 words, this shows that a small number of reviews are extremely long and detailed. The fact that the average (146 words) is higher than the median (93 words) confirms that these long reviews are pulling the mean upward.

Word Frequency Analysis

JK Rowling



Stephen King



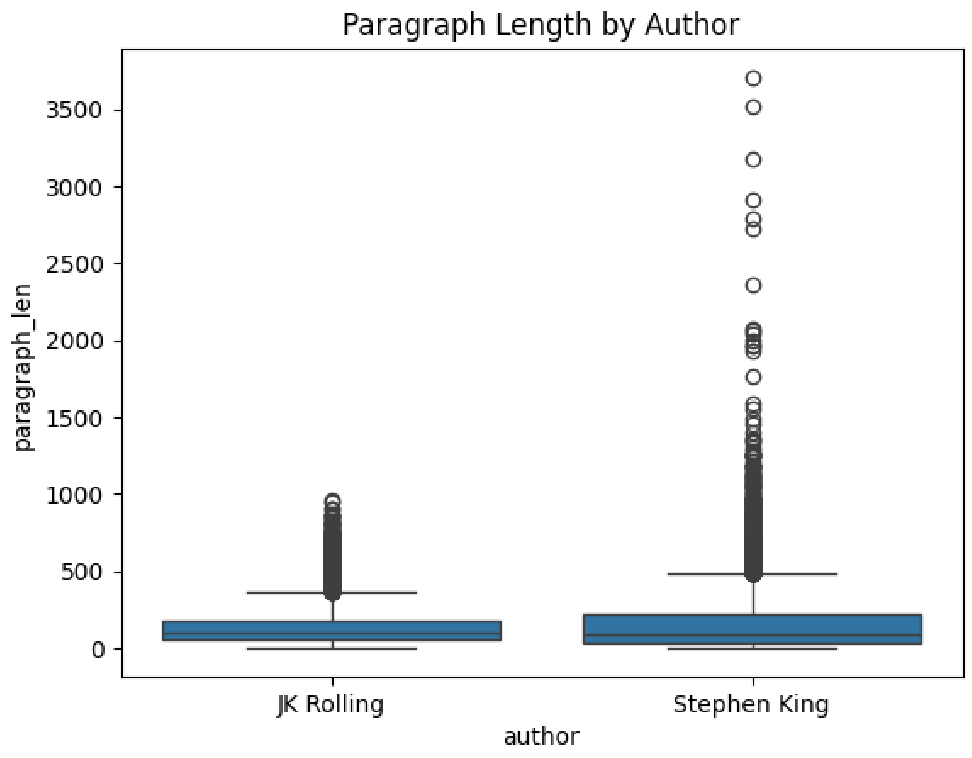
From these word clouds it was found that:

- Both authors share common words such as "the", "be", "a", "and", "to", "he", and "it". These words form the basic structure of English sentences for both writers.
- JK Rolling's writing shows distinctive use of character names like "Harry" and dialogue markers like "say". This suggests more character-driven storytelling with frequent dialogue.

- Stephen King's writing features more frequent use of "have" and "in", indicating potentially more descriptive passages and complex sentence structures.

Bivariate analysis

The bivariate analysis looked to expand the findings of the univariate analysis and made use of a boxplot.



- Based on the boxplot it is determined that Stephen King's paragraphs tend to be longer in length when compared to JK Rowling's paragraphs.

Modelling

For this section the LSTM model will be trained using the cleaned Pandas data frame.

Model preprocessing

In this step the focus was model preparation, so the data was split into training and test data, tokenised, sequenced and padded.

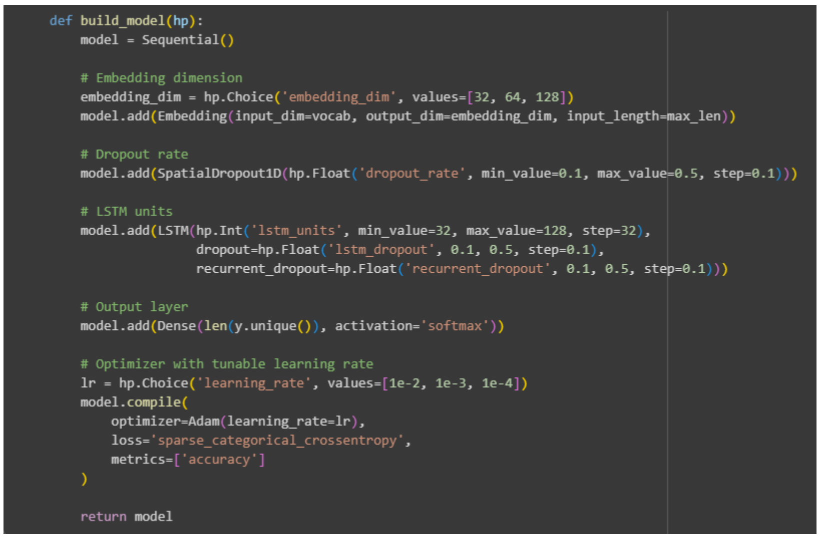
The data is first split into training (80%) and testing (20%) groups in order to prevent data leakage and overfitting. By following this split it also allows for an unbiased performance of model. Next the x training data is tokenised, meaning it is broken down into smaller units called tokens (Coursera, 2025). This is done so that the model is able to understand it. Finally, the training data is converted to sequences and then padding is applied to ensure that all samples are the same length. This step is critical as it ensures that every sentence fits to the model's expected input size (Sohail, 2024).

Model construction

This section will outline the steps taken to build, train and improve the model. An LSTM model will be constructed, a Long Short-Term Memory (LSTM) is a more advanced form of Hochreiter and Schmidhuber's Recurrent Neural Network (RNN) (GeeksForGeeks, 2025). LSTMs can detect long-term dependencies in sequential data, making them excellent for tasks such as language translation, speech recognition, and time series forecasting (GeeksForGeeks, 2025).

LSTM architecture

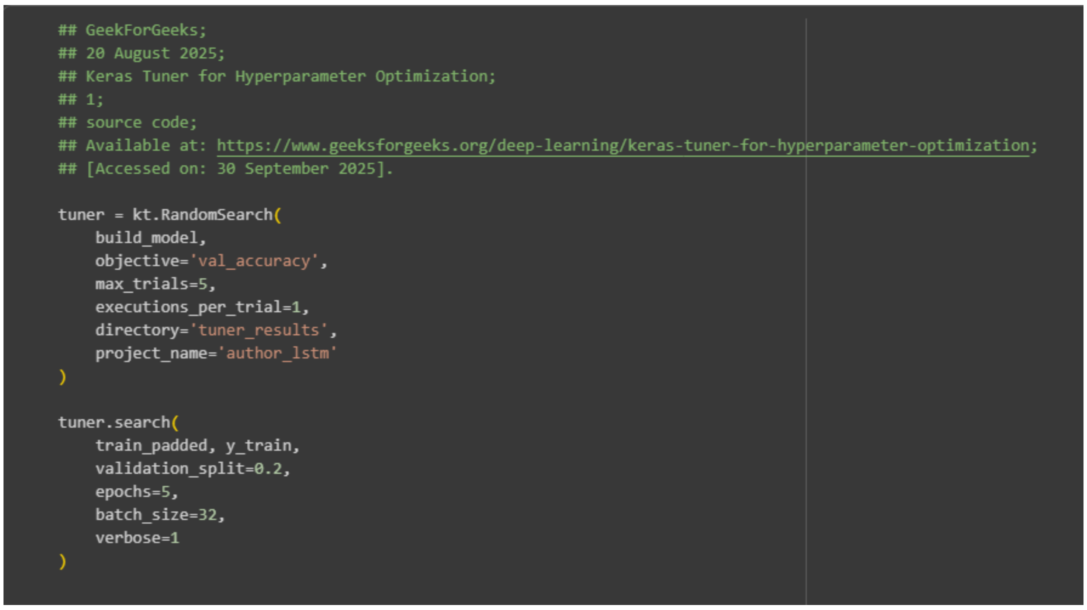
This model made use of an LSTM architecture which began with an Embedding layer to convert words into semantic vector representations, this was then followed by SpatialDropout1D for regularisation to minimise overfitting (Brownlee, 2022). The core LSTM layer then processes the input sequences, collect in a Dense output layer with Softmax activation for multi-class author classification.



```python
def build_model(hp):
    model = Sequential()

    # Embedding dimension
    embedding_dim = hp.Choice('embedding_dim', values=[32, 64, 128])
    model.add(Embedding(input_dim=vocab, output_dim=embedding_dim, input_length=max_len))

    # Dropout rate
    model.add(SpatialDropout1D(hp.Float('dropout_rate', min_value=0.1, max_value=0.5, step=0.1)))

    # LSTM units
    model.add(LSTM(hp.Int('lstm_units', min_value=32, max_value=128, step=32),
                   dropout=hp.Float('lstm_dropout', 0.1, 0.5, step=0.1),
                   recurrent_dropout=hp.Float('recurrent_dropout', 0.1, 0.5, step=0.1)))

    # Output layer
    model.add(Dense(len(y.unique()), activation='softmax'))

    # Optimizer with tunable learning rate
    lr = hp.Choice('learning_rate', values=[1e-2, 1e-3, 1e-4])
    model.compile(
        optimizer=Adam(learning_rate=lr),
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy']
    )

    return model
```

Hyperparameter tuning

One of the ways the model was improved was with the use of Hyperparameter tuning with the use of KerasTuner with Random Search. This allowed for the optimisation of embedding dimensions, LSTM units, dropout rates, and learning rate. Overall, hyperparameter tuning improved the generalisability of the model.



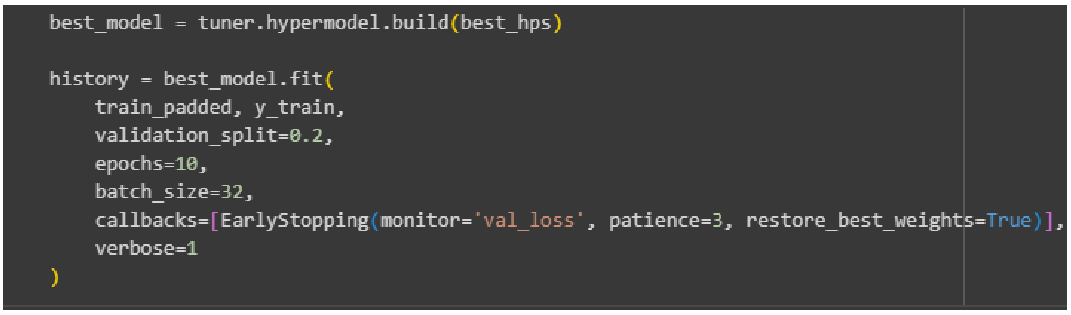
```
## GeekForGeeks;
## 20 August 2025;
## Keras Tuner for Hyperparameter Optimization;
## 1;
## source code;
## Available at: https://www.geeksforgeeks.org/deep-learning/keras-tuner-for-hyperparameter-optimization;
## [Accessed on: 30 September 2025].

tuner = kt.RandomSearch(
    build_model,
    objective='val_accuracy',
    max_trials=5,
    executions_per_trial=1,
    directory='tuner_results',
    project_name='author_lstm'
)

tuner.search(
    train_padded, y_train,
    validation_split=0.2,
    epochs=5,
    batch_size=32,
    verbose=1
)
```

Hyperparameter tuning found that these were the most optimal parameters:

Most optimal parameter	Value
Embedding dim	128
Dropout rate	0.2
LSTM units	64
LSTM dropout	0.2
Recurrent dropout	0.2
Learning rate	0.0001

Train model

Finally, the model was trained with the identified optimal parameters, incorporating EarlyStopping and the Adam optimiser to further prevent overfitting and ensure efficient convergence (GeeksForGeeks, 2025).

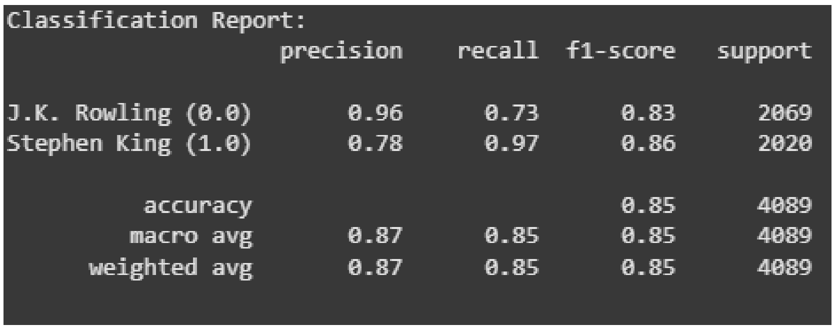


```
best_model = tuner.hypermodel.build(best_hps)

history = best_model.fit(
    train_padded, y_train,
    validation_split=0.2,
    epochs=10,
    batch_size=32,
    callbacks=[EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)],
    verbose=1
)
```

Evaluation

The evaluation of the model ran multiple tests to measure model performance, identify potential issues as well as areas for improvement. Tools like classification reports and confusion matrixes as well as ROC-AUC Curve were used to identify the best performing model.

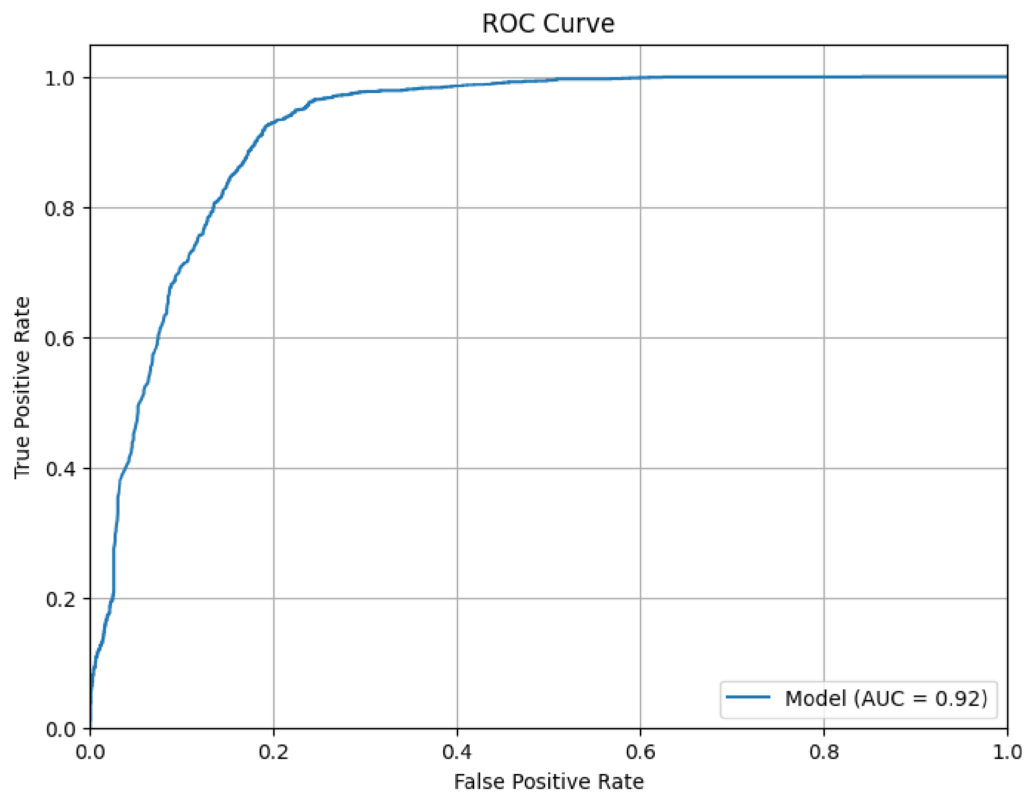
Classification report



```
Classification Report:
                    precision    recall  f1-score   support

J.K. Rowling (0.0)       0.96      0.73      0.83      2069
Stephen King (1.0)       0.78      0.97      0.86      2020

          accuracy                           0.85      4089
         macro avg       0.87      0.85      0.85      4089
      weighted avg       0.87      0.85      0.85      4089
```

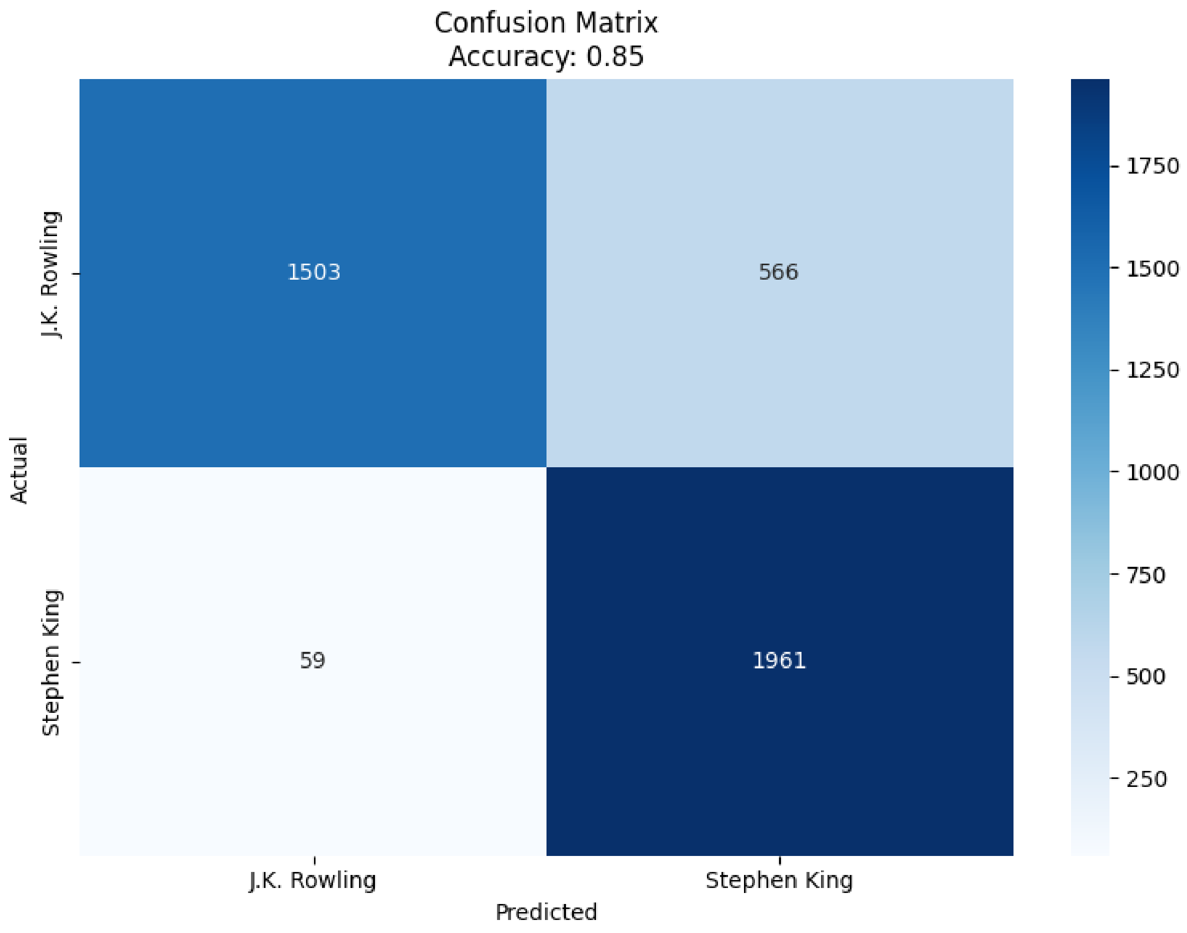
Based on the classification report it was found that the model shows a high performance across both authors, achieving 85% accuracy score overall. In terms of ‘JK Rolling’ (Class 0.0), the classification report showed a high precision at 96% but a more moderate recall at 73%. This means that while the prediction of the ‘JK Rolling’ class is highly reliable, the model misses 27% of the actual text. In contrast for ‘Stephen King’ (Class 1.0), the classification report showed a high recall at 97% but a more moderate precision at 78%. This means that the model is highly effective at detecting Stephen King’s writing style while being more selective but accurate in its JK Rolling predictions.

ROC-AUC curve



LSTM model shows excellent discriminative ability with an AUC of 0.92, meaning it is very effective at distinguishing between the authors classes.

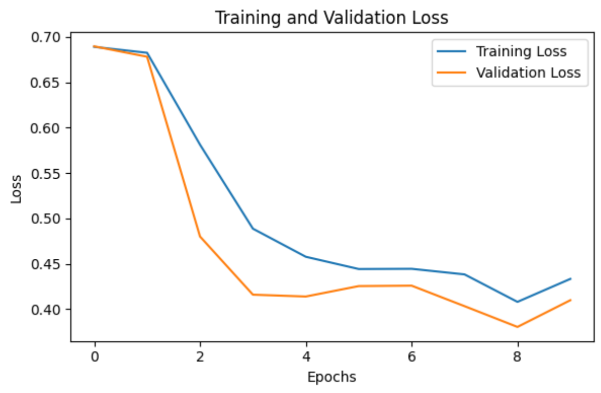
Confusion Matrix

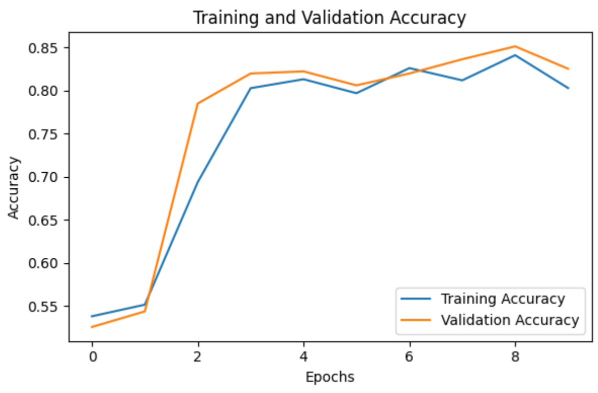


Author	Correct	Misclassified
JK Rolling	1503	566
Stephen King	1961	59

This further proves that the model is highly accurate when predicting Stephen King but does not perform as well with JK Rolling. However, it is important to take the recall and precision into consideration. By looking at both the confusion matrix and classification report it is concluded that the model is highly effective at detecting Stephen King's writing style while being more selective but accurate in its JK Rolling predictions.

Loss and accuracy





Loss Plot

Since, both the training and validation loss decrease at a steady rate, it indicates that model learns effectively.

Accuracy Plot

As found above from the classification report, the accuracy of the model reaches 85% however, in this plot the validation reaches slightly less. This indicates a minor overfitting. Overall, there are no major issues in this model, the model has a good performance with minor overfitting.

Conclusion

This project successfully demonstrated how NLP and machine learning can be applied to author classification based on text data. The model excelled at identifying Stephen King with a 97% recall and only 59 misclassifications. However, the model did struggle more with J.K. Rowling, achieving a 73% recall with 566 texts misclassified. Through preprocessing, exploratory analysis, and careful model construction, the pipeline achieved strong overall predictive performance of 85% accuracy.

Possible solutions to address the performance imbalance include applying class weighting during training or adjusting the maximum sequence length during text preprocessing. However, the current bias may be acceptable as the model still performs well overall. The console-based bot provides an interactive way for fans to engage with the book press's authors and explore the differences of writing style.

Future improvements could include expanding the dataset to cover more authors, experimenting with advanced deep learning architectures such as Transformers, and deploying the bot via a web interface for broader accessibility.

Reference List

Abdullah, S.M., 2024. *Text Preprocessing | NLP | Steps to Process Text*. [online] Available at: < https://www.kaggle.com/code/abdmental01/text-preprocessing-nlp-steps-to-process-text#11.-Lemmatization- > [Accessed: 29 September 2025]

Big Data Landscape., 2025. *Spark Machine Learning MLLib: StringIndexer*. [online] Available at: <https://medium.com/@big_data_landscape/spark-machine-learning-mllib-stringindexer-b91e3f6b8905>. [Accessed on: 27 Spetember 2025]

Brownlee, J., 2022. *Dropout with LSTM Networks for Time Series Forecasting*. [online] Available at: <https://machinelearningmastery.com/use-dropout-lstm-networks-time-series-forecasting>. [Accessed on: 27 Spetember 2025]

Coursera, 2025. *Tokenization in NLP: What Is It?*. [online] Available at: <https://www.coursera.org/articles/tokenization-nlp>. [Accessed on: 27 Spetember 2025]

GeeksForGeeks., 2025. *What is LSTM - Long Short Term Memory?*. [online] Available at: <https://www.geeksforgeeks.org/deep-learning/deep-learning-introduction-to-long-short-term-memory>. [Accessed on: 27 Spetember 2025]

GeekForGeeks., 2025. *ML | Overview of Data Cleaning*. [online] Available at: <https://www.geeksforgeeks.org/data-cleansing-introduction/>. [Accessed on: 29 September 2025]

GeekForGeeks., 2025. *EDA | Exploratory Data Analysis in Python*. [online] Available at: <https://www.geeksforgeeks.org/exploratory-data-analysis-in-python/>. [Accessed on: 29 September 2025]

GeeksForGeeks., 2025. *Using Early Stopping to Reduce Overfitting in Neural Networks.* [online] Available at: <https://www.geeksforgeeks.org/deep-learning/using-early-stopping-to-reduce-overfitting-in-neural-networks>. [Accessed on: 27 Spetember 2025]

Sohail, S., 2024. Understanding Padding in NLP: Types and When to Use Them. [online] Available at: <https://saadsohail5104.medium.com/understanding-padding-in-nlp-types-and-when-to-use-them-bacae6cae401> [Accessed 30 September 2025]