**Final Term Project Report**

* **Student Name**: Idowu Dolapo  
  **Project Title**: Machine Learning Classifier Comparison on Amazon Reviews  
  **Dataset**: Amazon – "Effective Java" (Synthetic)  
  **Course**: CS 634 <854> Data mining   
  **Date**: 04/03/2025

Implementation and Code Usage

**Machine Learning Classifier Comparison on Amazon Book Revies**

**Abstract:**

This project compares the performance of three machine learning models Random Forest, Naive Bayes, and LSTM in classifying whether an Amazon review is related to the book *Effective Java*. Because a specific dataset wasn’t available, the student created a small, synthetic dataset of 10 manually labeled reviews. This allowed a controlled environment to evaluate the models under consistent conditions.

To measure performance, 10-fold cross-validation was used for the traditional classifiers, while the LSTM model was evaluated using a validation split. Key metrics like precision, recall, F1 score, false positive rate, and false negative rate were calculated manually. The results provide insight into how each model handles text classification with limited data, showcasing the practical use of machine learning and natural language processing techniques.

**Introduction:**

In today’s data-driven world, the ability to extract meaningful insights from unstructured text is more valuable than ever. This project explores and compares the performance of three machine learning classifiers Random Forest, Naive Bayes, and a deep learning-based LSTM (Long Short-Term Memory) network on a dataset of Amazon product reviews. The primary objective is to determine whether a given review pertains to the widely referenced programming book “*Effective Java”*.

To achieve this, we apply fundamental natural language processing (NLP) techniques to preprocess the textual data, followed by training and evaluating the three models on the same dataset. A manually labeled synthetic dataset was used to simulate a real-world classification task in a controlled environment. To ensure robust evaluation, 10-fold cross-validation was implemented for the traditional classifiers, while a train-validation split was used for the LSTM model. In addition, we manually computed key performance metrics such as precision, recall, F1-score, and error rates to gain deeper insights into each model's effectiveness.

By the end of this project, we aim to understand how different types of classifiers ranging from probabilistic models to deep neural networks handle short form review text, and which model performs best under these specific conditions.

**Core Concept and Principles**

Several key concepts were applied throughout the project to ensure a structured and meaningful analysis.

* **Text Preprocessing**: Raw review text was transformed into a format suitable for machine learning models. For traditional classifiers, TF-IDF (Term Frequency–Inverse Document Frequency) was used to convert text into numerical vectors. For the LSTM model, reviews were tokenized and padded into equal length sequences.
* **Model Selection**: Three classifiers were chosen for their distinct learning approaches—Random Forest (ensemble learning), Naive Bayes (probabilistic modeling), and LSTM (deep learning with sequence awareness). The models were implemented using Python libraries such as Scikit-learn and TensorFlow.
* **Cross-Validation**: To test model consistency and prevent overfitting, 10-fold cross-validation was used for Random Forest and Naive Bayes. This method helps validate performance across multiple data splits.
* **Manual Metric Evaluation**: Performance was assessed by manually calculating key metrics such as True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). From these values, secondary metrics like Precision, Recall, F1 Score, False Positive Rate (FPR), and False Negative Rate (FNR) were derived to provide a deeper understanding of each model's accuracy and reliability.

**Dataset Description**

The project followed a simple but effective workflow, starting with building a custom dataset. Since there was not a pre-made dataset available for this specific task, a small set of 10 reviews was manually created. Each review was labelled as either related (1) or unrelated (0) to the book Effective Java.

If the CSV file is not found when running the program, the script is designed to automatically generate the same dataset within the code. This makes the project easy to run without needing to manually prepare any files in advance.

Example rows from the dataset:

|  |  |
| --- | --- |
| **Review** | **Label** |
| "This book is a must-read for Java developers." | 1 |
| "Terrible quality, nothing useful." | 0 |
| "Effective Java helped me write cleaner code." | 1 |
| "Not about Java at all, misleading title." | 0 |

This small yet meaningful dataset helps us validate the effectiveness of each classifier in a controlled environment.

**Project workflow:**

The project followed a simple but effective workflow, moving from building the dataset to evaluating how each model performed.

* **Preparing the Data**  
  The review text was first converted into a format the models could process. For the Random Forest and Naive Bayes classifiers, TF-IDF was used to transform the text into numerical vectors. For the LSTM model, the text was tokenized and padded so that all sequences were of equal length—an essential step when working with deep learning models.
* **Training the Models**  
  The Random Forest and Naive Bayes models were trained using 10-fold cross-validation in Scikit-learn. This approach helps ensure reliable results by evaluating the models across multiple data splits. The LSTM model was developed using TensorFlow/Keras and trained on 80% of the dataset, with the remaining 20% used for validation.
* **Evaluating Performance**  
  After the models generated predictions, a custom Python function was used to manually calculate key performance metrics. These included precision, recall, F1 score, false positive rate, and false negative rate. The metrics helped in comparing how each model performed on the same classification task.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Precision | Recall | F1 Score | FPR | FNR |
| Random Forest | 1.00 | 1.00 | 1.00 | 0.00 | 0.00 |
| Naive Bayes | 1.00 | 1.00 | 1.00 | 0.00 | 0.00 |
| LSTM | 1.00 | 1.00 | 1.00 | 0.00 | 0.00 |

**Project Requirements:**

The project was completed in accordance with the following requirements:

* Implementation of three classifiers: Random Forest, Naive Bayes, and LSTM
* Use of 10-fold cross-validation for Random Forest and Naive Bayes to ensure reliable evaluation
* Manual calculation of performance metrics, including: True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN), False Positive Rate (FPR), False Negative Rate (FNR), Precision, Recall, and F1 Score
* Development and use of a synthetic dataset, as instructed
* Submission of a detailed report outlining the process, implementation, and results

**Step by Step Implementation:**

* **Load Data**  
  The script begins by checking for a CSV file containing the labeled reviews. If it’s not found, a predefined synthetic dataset is generated in memory to ensure smooth execution.
* **Preprocess Text**  
  TF-IDF is used to transform the text into numerical vectors for Random Forest and Naive Bayes. For the LSTM model, the text is tokenized and padded to a consistent length.
* **Train Models**  
  Random Forest and Naive Bayes are trained using 10-fold cross-validation in Scikit-learn. The LSTM model is built using TensorFlow/Keras and trained with an 80/20 train-validation split.
* **Predict and Evaluate**  
  Once training is complete, the models generate predictions, and a custom function is used to manually calculate metrics like precision, recall, F1 score, FPR, and FNR.

**Conclusion:**

This project served as a concise yet effective demonstration of machine learning techniques for text classification. The Random Forest, Naive Bayes, and LSTM models were successfully implemented and each achieved perfect results on a small, manually created dataset. While this indicates that the models functioned correctly and were evaluated as intended, it also emphasizes a key limitation—the dataset’s simplicity and size.

To truly assess model performance in a real-world scenario, larger and more diverse datasets would be necessary. Despite this constraint, the project effectively reinforced core concepts such as data preprocessing, model training, and manual metric evaluation, providing valuable hands-on experience with natural language processing and supervised learning.

**Screenshots:**

Figure 1: Screenshot of program execution showing classification output.  
Figure 2: Screenshot showing metrics output for each model.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Other:**

* Source Code: classification\_project.py
* Dataset: amazon\_effective\_java\_dataset.csv
* README Instructions: README.md
* Report Document: Final Term Project Report.docx
* Screenshots: Program running screenshot.png, Program running screenshot2.png

Are included in the ZipFileFolder: Idowu\_Dolapo\_Finaltermproj.zip

**GitHub Repository:**  
<https://github.com/Dolapo-22/finalproject.git>