Natural Language Processing (CS5803): Assignment-2
Text Classification

Rachit Deshpande (AI24MTECH11003)
Saurabh Gangwar (AI24MTECH11008)
Aditya Shinde (AI24MTECH11004)

March 1, 2025

1 Dataset

In this assignment, we work with two different datasets for text classification:

Dataset 1: This dataset consists of research article metadata, including the *ID*, *Title*, and *Abstract*, along with labels for multiple categories such as Computer Science, Physics, Mathematics, Statistics, Quantitative Biology, and Quantitative Finance. The task is to classify the articles based on their title and abstract.

Dataset 2: This dataset consists of short textual content labeled with domains such as Entertainment, Healthcare, Sports, Technology, and Tourism. The classification task is to predict the correct domain based on the content.

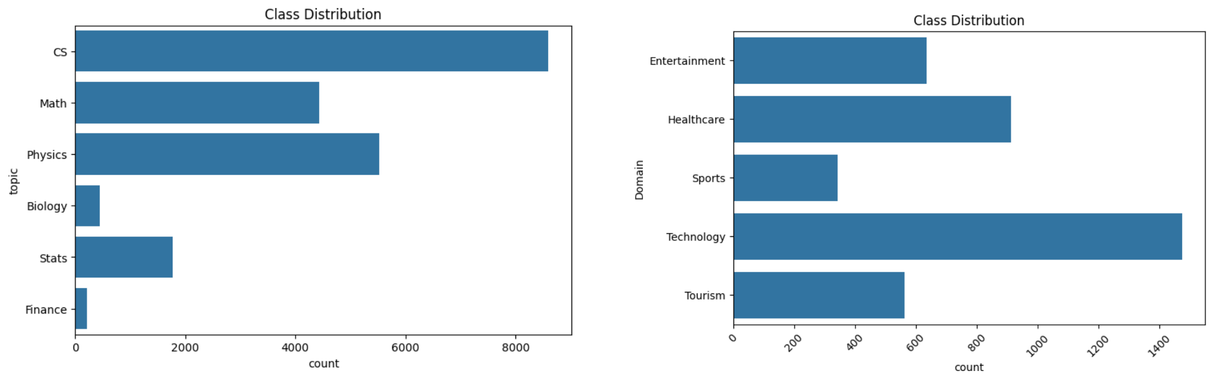

Figure 1: Distribution of Target Variables in Dataset 1 (Left) and Dataset 2 (Right)

2 Methodology

In this study, we approach the problem of text classification using four different models: Random Forest, Support Vector Machine (SVM), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM). The methodology consists of dataset preprocessing, feature extraction, model training, and evaluation.

2.1 Dataset Preprocessing

We begin by loading the dataset, which contains article titles, abstracts, and topic labels. Since the dataset follows a multi-label format, we convert it into a single-label classification problem by assigning each instance to its most dominant topic. The six topic categories include CS, Physics, Math, Stats, Biology, and Finance.

To standardize the input for our models, we preprocess the text by combining the title and abstract, removing stop words, and performing tokenization. Additionally, the topic labels are encoded into numerical values using LabelEncoder.

The dataset is then split into training (50%), validation (20%), and testing (30%) sets, ensuring a balanced distribution of classes. This split allows effective hyperparameter tuning on the validation set while preserving an unseen test set for final evaluation.

2.2 Feature Extraction

We employ two different text vectorization techniques depending on the model type:

- **Traditional Machine Learning Models:** For the Random Forest and SVM classifiers, we use TF-IDF Vectorization. The TfidfVectorizer is applied with a vocabulary size of 1000 and English stop words removed. This method converts text data into a numerical feature matrix, emphasizing important words based on their relative frequency.
- **Neural Network Models:** For RNN and LSTM, we use tokenization and sequence padding. The Tokenizer from Keras is employed to convert text into sequences of integers, and pad_sequences ensures uniform input lengths. The maximum sequence length is set to 100 to balance information retention and computational efficiency.

2.3 Model Training and Evaluation

2.3.1 Random Forest

A RandomForestClassifier is trained with hyperparameter tuning using GridSearchCV. We optimize n_estimators (50, 100), max depth (None, 10), and min samples split (2, 5). The best model is selected based on the highest weighted F1-score.

2.3.2 Support Vector Machine

We train an SVM model using a linear kernel with hyperparameter tuning for the regularization parameter C (1, 10). Grid search is performed with 3-fold cross-validation, optimizing for the weighted F1-score.

2.3.3 Recurrent Neural Network

A simple RNN model is built using an embedding layer (input_dim=1000, output_dim=16) followed by a SimpleRNN layer with 32 units, a dropout layer (0.5), and a dense softmax output layer. The model is trained using categorical cross-entropy loss with the Adam optimizer.

To handle class imbalance, we compute class weights and apply them during training.

2.3.4 Long Short-Term Memory (LSTM)

The LSTM model follows a similar architecture to the RNN, replacing the SimpleRNN layer with an LSTM layer of 64 units. LSTMs capture long-term dependencies in text, making them effective for classification tasks. The training process remains the same, using categorical cross-entropy loss and class weights.

2.4 Performance Metrics

All models are evaluated using:

- **Accuracy:** Measures the overall correctness of predictions.
- **F1-score:** A weighted average of precision and recall.
- **ROC-AUC Score:** Evaluates classification performance across different thresholds.
- **Average Precision:** Summarizes precision-recall trade-offs.
- **Confusion Matrix:** Visualizes classification errors across categories.

Hyperparameter tuning and model training are performed using the training and validation sets, while final performance is reported on the test set.

3 Models Used

In this assignment, we investigate four machine learning and deep learning models for text classification: Random Forest, Support Vector Machine (SVM), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM). We choose each model based on its effectiveness in handling textual data, and the details of its implementation are given below.

3.1 Random Forest

Random Forest is an ensemble algorithm that builds multiple decision trees in training and provides the class with the most votes. This has the effect of reducing overfitting, increasing generalization, and being extremely interpretable for text classification.

3.1.1 Implementation Details

We use Random Forest classifier implemented by Scikit-learn's RandomForestClassifier. The features in the input are obtained through Term Frequency-Inverse Document Frequency (TF-IDF) vectorization, which turns text data into a numerical value.

The model is trained using the following hyperparameters:

- Number of trees (n estimators): 100
- Max depth (max depth): 10
- Minimum samples per split: 5

We conduct the hyperparameter tuning with GridSearchCV to identify the best values that provide better classification performance.

3.2 Support Vector Machine (SVM)

Support Vector Machines are efficient for text data classification of high dimension. The SVM model tries to identify the best hyperplane that maximizes class margin, so it is a good option for binary and multi-class classification.

3.2.1 Implementation Details

The SVM classifier is applied via Scikit-learn's SVC (Support Vector Classifier). TF-IDF representation is utilized as input, the same as with the Random Forest model.

We employ the following hyperparameters:

- **Kernel: Linear**
- **Regularization parameter (C): 1.0**

Since text data is generally linearly separable in high-dimensional TF-IDF space, we employ a linear kernel to obtain optimal classification performance.

3.3 Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNNs) are intended to process sequential data. They retain hidden states across time steps, allowing the model to capture context information in text classification problems.

3.3.1 Implementation Details

The RNN model is executed with Keras and TensorFlow as the backend. The text data is pre-processed using tokenization, encoded into integer sequences, and padded to a common length. The architecture of the model includes:

- **Embedding layer: Maps words to dense vector representations.**
- **Basic RNN layer: 32 recurrent units to extract sequential dependencies.**

- **Dropout layer: 0.5 dropout to avoid overfitting.**
- **Dense output layer: Softmax activation for multi-class classification.**

We train the model with categorical cross-entropy loss and Adam optimizer, a batch size of 32 and 10 training epochs.

3.4 Long Short-Term Memory (LSTM)

LSTMs are a more sophisticated type of RNN that use gating mechanisms to control long-term dependencies, which are useful for extracting contextual meaning in long texts.

3.4.1 Implementation Details

The LSTM model mimics the same structure as RNN but substitutes the SimpleRNN layer with an LSTM layer:

- Embedding layer: Encodes words to 16-dimensional dense vectors.
- LSTM layer: 64 memory units for capturing long-range dependencies.
- Dropout layer: Dropout rate of 0.5 for regularization.
- Dense output layer: Softmax activation for classification.

We employ categorical cross-entropy loss and the Adam optimizer with a learning rate of 0.001. The model is trained for 10 epochs with batch normalization to stabilize training.

3.5 Comparison of Models

Each model has unique strengths:

- **Random Forest is fast and interpretable but lacks deep contextual understanding.**
- **SVM works well with TF-IDF features but has difficulty with big datasets.**
- **RNN captures sequential dependencies but may suffer from vanishing gradients.**
- **LSTM reduces vanishing gradients and performs well in long-text classification**.

This comparative approach provides insights into both traditional and deep learning methods for text classification.

4 Results & Analysis

In this section, we evaluate the performance of our models using various metrics such as accuracy, F1-score, ROC-AUC, and Average Precision. We also present confusion matrices for a detailed performance breakdown.

4.5 Dataset 1 Results

Table 1 summarizes the performance metrics of all four models on Dataset 1.

Model	**Accuracy**	**F1-score**	**ROC-AUC**	**Avg. Precision**
Random Forest	0.7614	0.7259	0.9117	0.5967
SVM	0.7161	0.7343	0.9290	0.6181
RNN	0.5417	0.5854	0.7851	0.4348
LSTM	0.4994	0.5032	0.8618	0.4888

Table 1: Performance Metrics for Dataset 1

Figures 2 to 5 show the confusion matrices for each model.

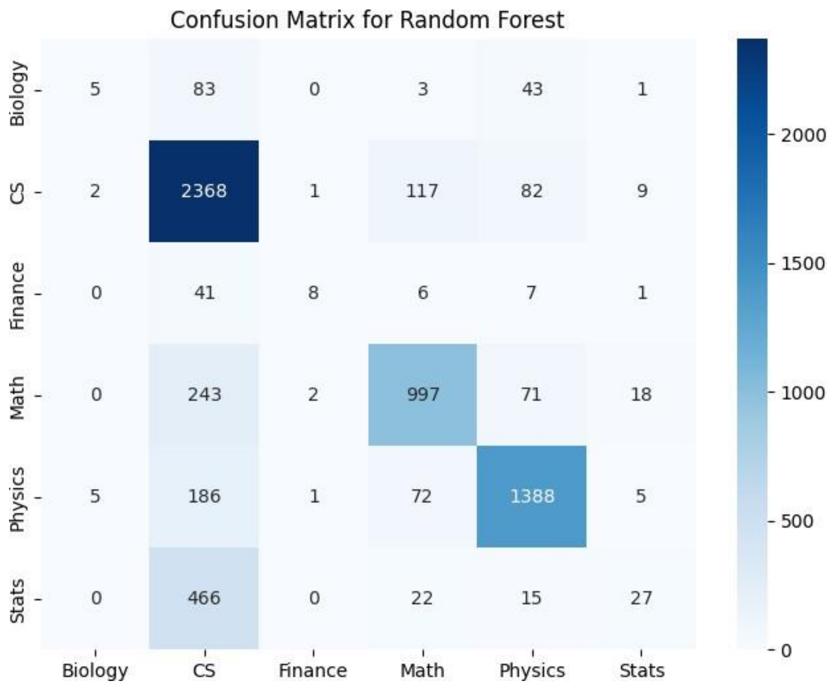

Figure 2: Confusion Matrix for Random Forest on Dataset 1

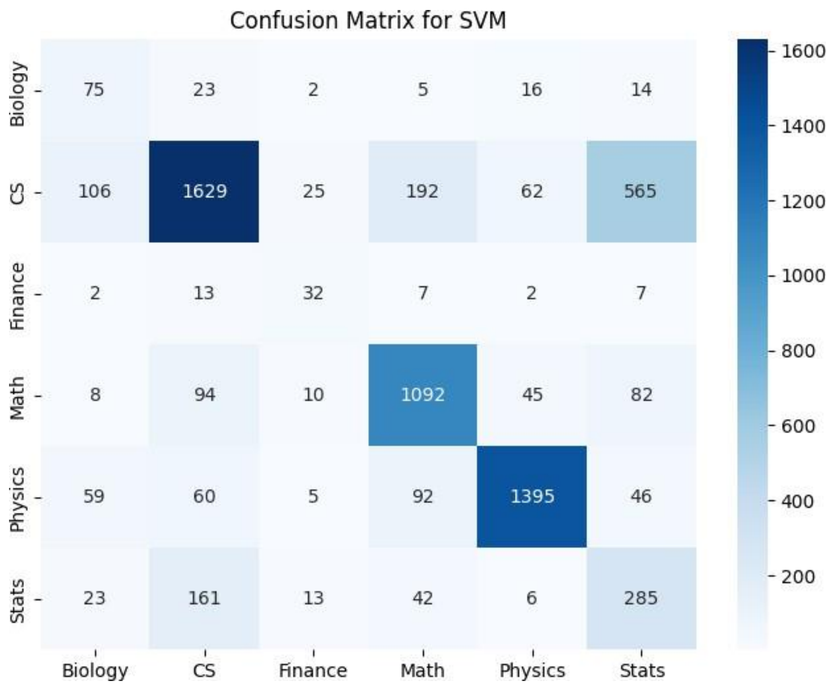

Figure 3: Confusion Matrix for SVM on Dataset 1

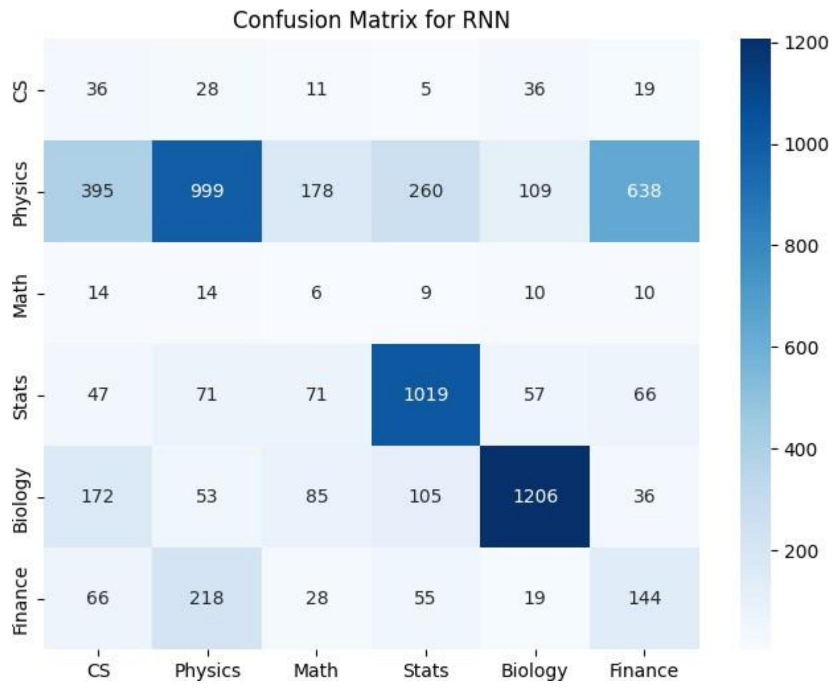

Figure 4: Confusion Matrix for RNN on Dataset 1

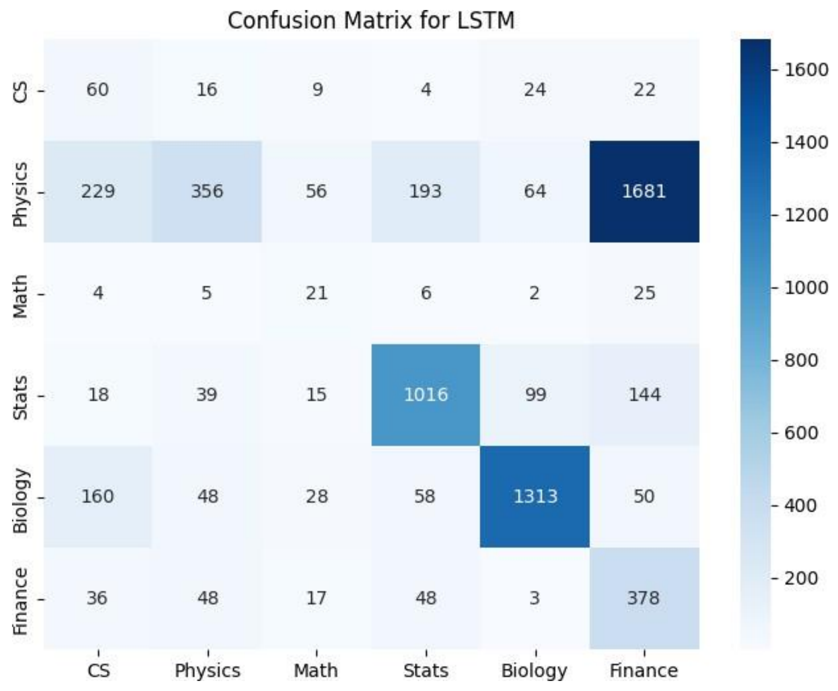

Figure 5: Confusion Matrix for LSTM on Dataset 1

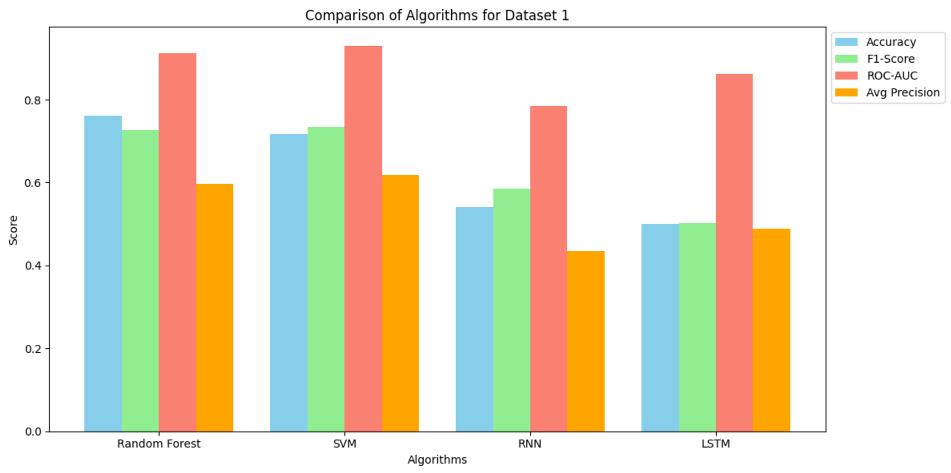

Figure 6: Model Performance Comparison for Dataset 1

4.6 Dataset 2 Results

Table 2 summarizes the performance metrics of all four models on Dataset 2.

Model	Accuracy	F1-score	ROC-AUC	Avg. Precision
Random Forest	0.9949	0.9949	0.9999	0.9996
SVM	0.9966	0.9966	1.0000	0.9999
RNN	0.8364	0.8415	0.9708	0.8533
LSTM	0.9890	0.9891	0.9996	0.9987

Table 2: Performance Metrics for Dataset 2

Figures 7 to 10 show the confusion matrices for each model.

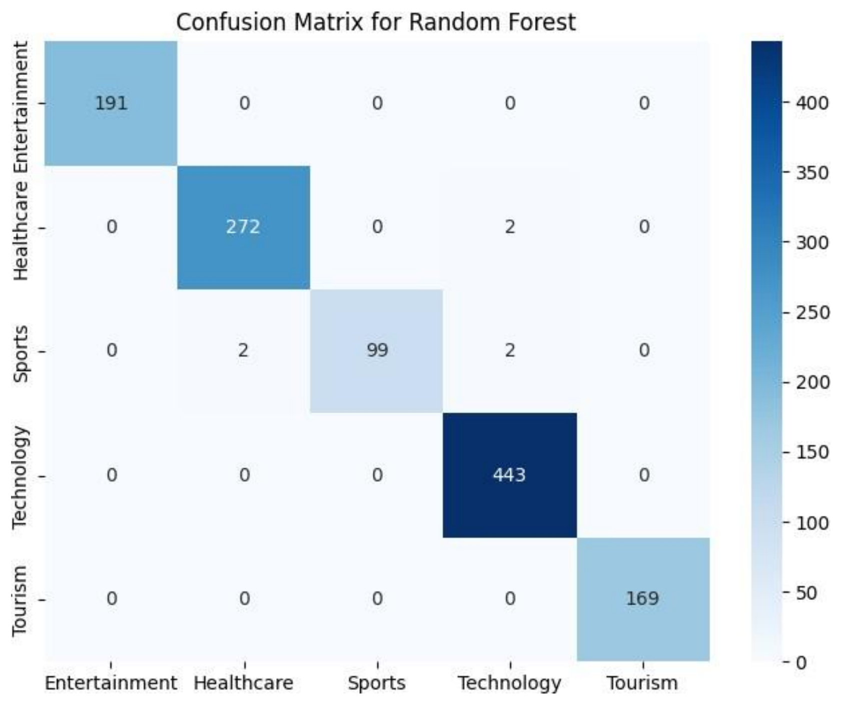

Figure 7: Confusion Matrix for Random Forest on Dataset 2

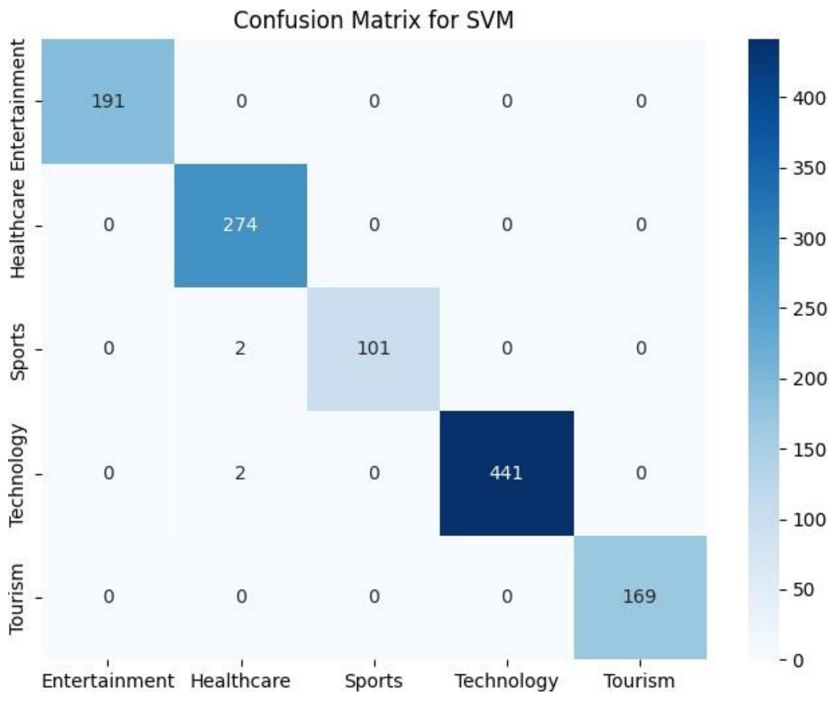

Figure 8: Confusion Matrix for SVM on Dataset 2

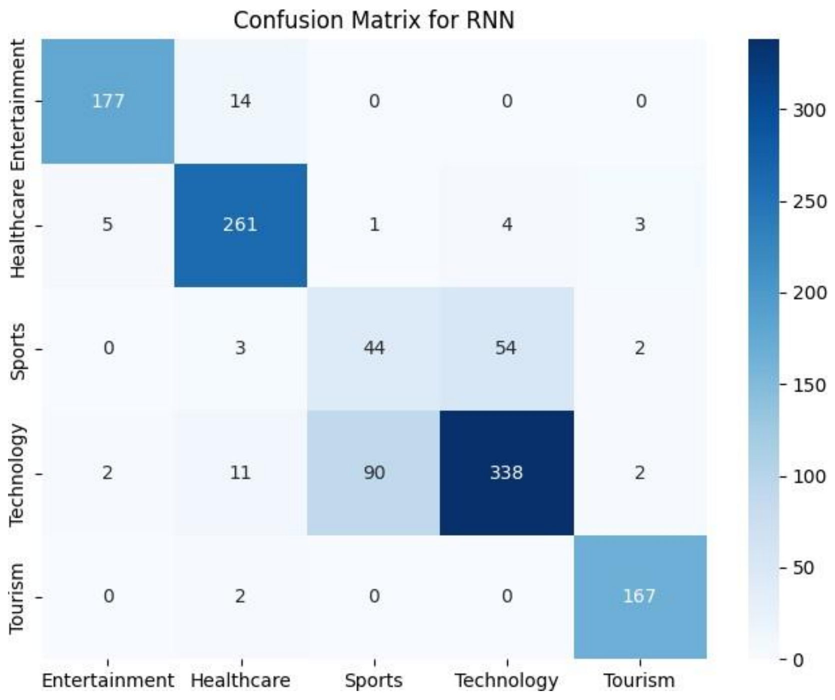

Figure 9: Confusion Matrix for RNN on Dataset 2

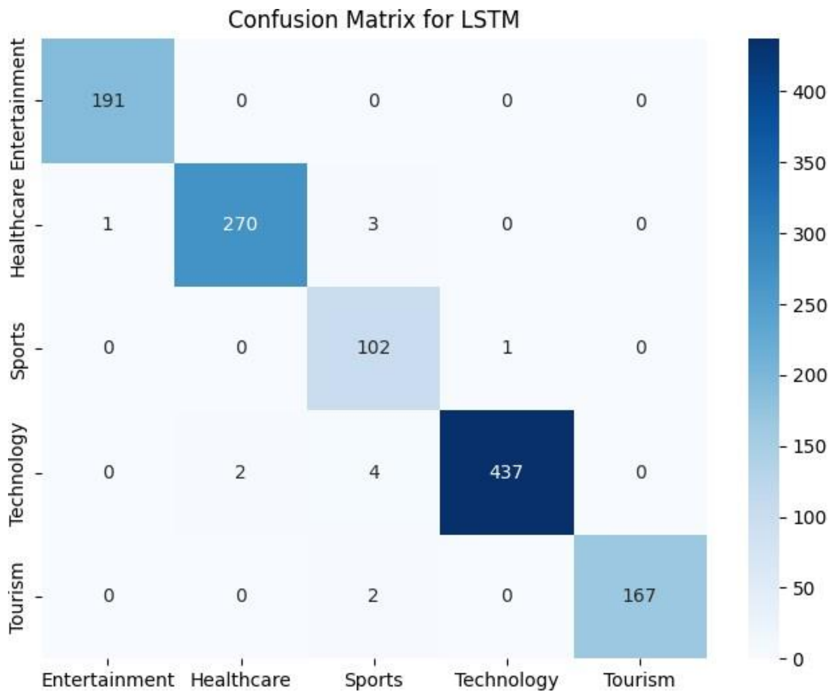

Figure 10: Confusion Matrix for LSTM on Dataset 2

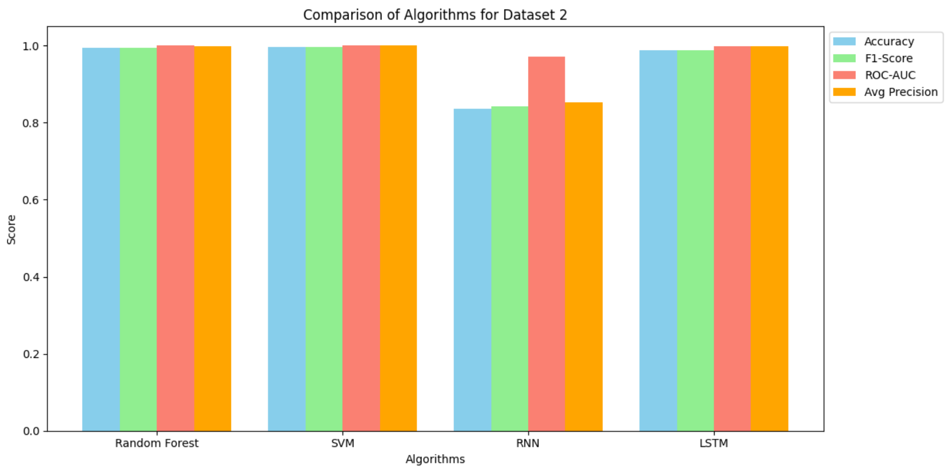

Figure 11: Model Performance Comparison for Dataset 2

4.7 Analysis of Results

From the results, we observe significant differences in model performance across the two datasets:

- **Dataset 1:** Traditional machine learning models (Random Forest and SVM) outperform deep learning models (RNN and LSTM). SVM achieves the highest F1-score (0.734), while Random Forest achieves the best ROC-AUC score (0.911).
- **Dataset 2:** All models perform exceptionally well, with SVM and Random Forest achieving near-perfect classification. LSTM also performs well, with a test accuracy of 0.989.
- **Comparison:** Deep learning models (RNN and LSTM) struggle with Dataset 1 but excel in Dataset 2, suggesting that the complexity of dataset structure influences performance.
- **Confusion Matrices:** The matrices indicate that classes like 'CS' and 'Physics' are better classified, while classes with fewer samples (e.g., 'Finance') have lower recall.

These insights suggest that while deep learning methods require more data for generalization, traditional models like SVM can be effective when data is limited.