**1. Introduction**

Fake news has become an increasingly pervasive issue in today's digital age, as the rapid dissemination of information through online platforms has made it easier for misleading or entirely fabricated news stories to spread. The spread of fake news can have far-reaching consequences, including the potential to influence public opinion, sway political discourse, and even incite violence. Therefore, the development of effective fake news classification algorithms has gained significant importance in recent years [1][2].

To address this challenge, researchers have turned to machine learning and natural language processing techniques to build automated systems capable of discerning between genuine and fake news articles [3]. These systems leverage a wide array of features, including textual content, metadata, and social network data, to make accurate classifications. This burgeoning field has seen substantial progress, with various methodologies and approaches being explored and refined.

By critically examining the methodologies and evaluation results, we hope to shed light on the field of fake news classification, and the avenues for future research and development.

* 1. Motivation

In today's hyperconnected world, the phenomenon of fake news has cast a long and menacing shadow over the information landscape. The ease with which fabricated or misleading information can be disseminated through various online channels has given rise to a pressing concern that transcends borders and political boundaries. The consequences of fake news are profound, with the potential to manipulate public opinion, disrupt democratic processes, and even incite violence. Consequently, the development of robust and reliable fake news classification systems has become an imperative undertaking for researchers, policymakers, and technologists alike [4].

* 1. Relevance

The use of ML and LSTM models to classify fake news will help many organizations and people, it will reduce the time for them to detect this.

**2. Literature review**

Fake news, characterized by the dissemination of false or misleading information disguised as legitimate news, has emerged as a critical issue in the digital age [5]. Detecting and classifying fake news has become a primary focus for researchers and practitioners alike [2].

Shu et al. (2017) laid the groundwork for fake news classification, emphasizing the significance of text-based features, user behavior, and network structures [6].

Ruchansky et al. (2017) introduced a hybrid deep model for fake news detection, highlighting the relevance of deep learning techniques [7].

Volkova et al. (2017) conducted linguistic analysis to identify deception in online dating profiles, emphasizing the role of linguistic cues and stylistic patterns [8].

Recent research by Wu et al. (2021) introduced a multimodal framework for fake news classification, integrating textual and visual information [9].

Ethical considerations in fake news classification have gained prominence, emphasizing the importance of avoiding censorship and maintaining transparency [10].

Challenges remain in the field, including the emergence of deepfake content and the need for ongoing adaptation to evolving tactics [11].

In conclusion, the literature on fake news classification combines techniques from data mining, machine learning, linguistics, and ethics, reflecting a dynamic field that continues to evolve.

**3. Methodology**

This section describes the details about the steps we have done to classify the news from the scratch. This part has details about the collection of the dataset, cleaning of the data and also the architecture of models I have used.

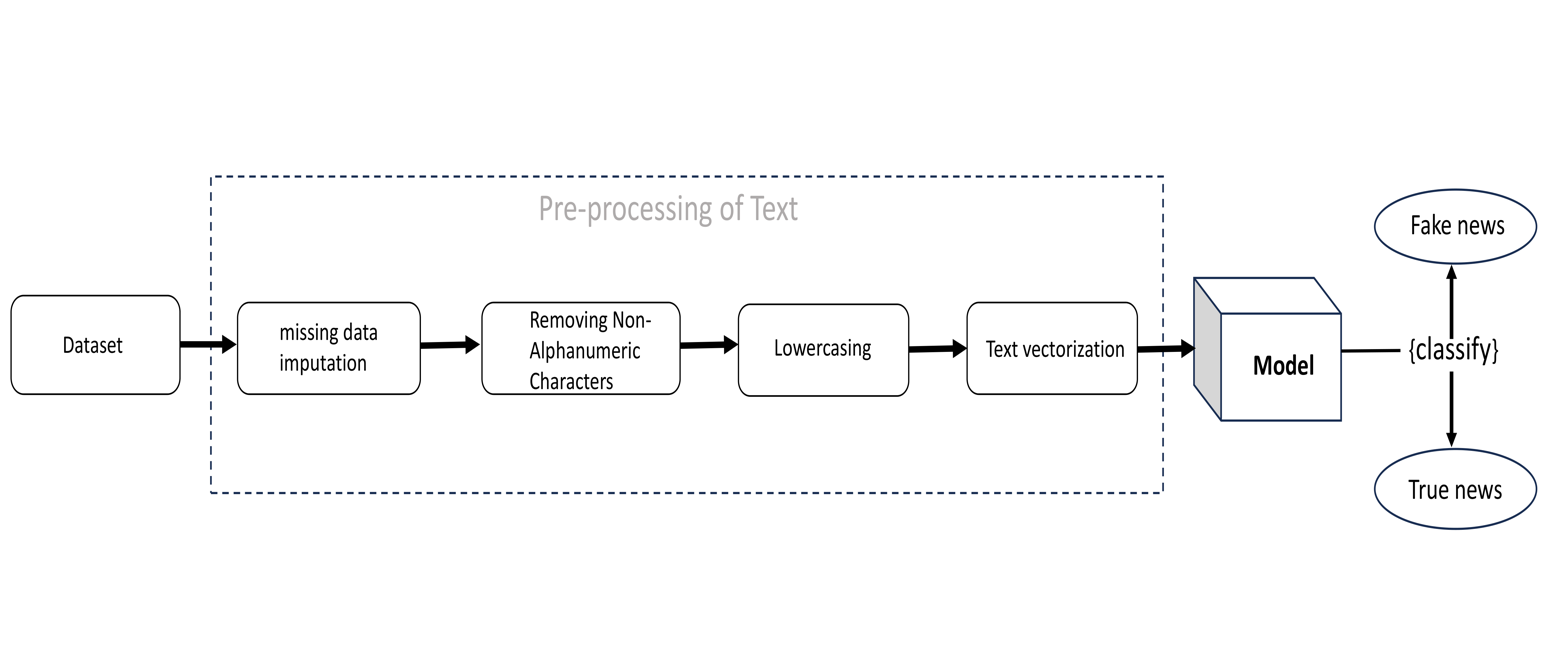


Fig 1: Stepwise proposed methodology

Step 1: At first, we have created a dataset by collecting some of the fake news datasets from internet and merged them all in a single csv file.

Step2: This step is about the preprocessing of the data –   
2.1 At first, we have removed the rows of the dataset which contains missing values in at least one of the columns. If there are still any missing values present then we have replaced it by filling it with empty string.

2.2 Secondly, we have removed all the non-alphanumeric characters from our text data because they do not provide any importance to the model.

2.3 Then we lower cased the words so that it has the uniformity in data and it also reduces the dimension of the unique words which is good for any model.

2.4 Then we have done text vectorization also known as text feature extraction by using count vectorizer (bag of words). It converts the text into numbers, so that the model can understand this. We have used Tokenizer to convert string to numbers for the inputs in deep learning model.

Step3: The models we have used and their details:

3.1 Multinomial Naïve Bayes:

Multinomial Naive Bayes is a variant of the Naive Bayes algorithm, which is based on Bayes' theorem. It's called "naive" because it makes a simplifying assumption that the features (in this case, words or tokens in text data) are conditionally independent given the class label. This assumption, though often not entirely true in real-world text data, simplifies the computation and still produces good results for many NLP tasks. Multinomial Naive Bayes is specifically designed for discrete data, such as word counts in text documents. It assumes that the features are distributed according to a multinomial distribution, which is a generalization of the binomial distribution for multiple categories (words in this context).

* P(x|C) = (Number of times word x appears in documents of class C) / (Total number of words in documents of class C)

The likelihood of observing a specific word (x) in a document given a class (C). This is typically modelled as a multinomial distribution

3.2 Random Forest:

Random Forest is a highly effective ensemble learning algorithm that harnesses the collective wisdom of multiple decision trees to make predictions with remarkable accuracy and resilience. It accomplishes this by introducing two critical elements of randomness: random sampling of data points and random selection of features, which significantly mitigates the risk of overfitting, ensuring that it can handle complex and noisy datasets with aplomb.

This algorithm's prowess extends across a spectrum of applications, spanning classification and regression tasks, where it consistently delivers dependable results. Its ability to assess the importance of individual features in making predictions provides valuable insights for feature selection and data understanding.

Random Forest's reputation for versatility and robustness has made it a stalwart choice in the machine learning toolkit, capable of tackling challenges across various domains and accommodating large datasets, thanks to its potential for parallelization. While its default settings are user-friendly, its performance can be fine-tuned through parameter optimization to cater to specific problem nuances, making it a cornerstone in modern machine learning.

3.3 Bidirectional LSTM:

Bidirectional LSTM (BiLSTM) is a variant of the Long Short-Term Memory (LSTM) recurrent neural network architecture. It processes sequential data in both forward and backward directions, capturing context from both past and future elements in the sequence. BiLSTM is particularly effective in tasks where understanding context from both directions is essential, such as natural language processing tasks like sentiment analysis and machine translation. During training, it updates its hidden states and cell states using information from both directions, making it capable of capturing long-range dependencies in the data. This bidirectional context modelling helps improve performance in tasks requiring a comprehensive understanding of sequential data. However, it can be computationally intensive compared to unidirectional LSTMs and may not be suitable for all applications.

3.4 BERT model:

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a natural language processing (NLP) model developed by Google in 2018. It's one of the most influential and widely used NLP models, known for its ability to understand the context of words in a sentence and provide pre-trained contextual embeddings for a wide range of NLP tasks.

Step 4: Model Evaluation steps with details:

The model performance is measured based on accuracy, precision, recall, F1-score, ROC curve.

Let’s define some terms-

* True Positive (TP): The number of correctly predicted positive instances.
* True Negative (TN): The number of correctly predicted negative instances.
* False Positive (FP): The number of negative instances incorrectly predicted as positive (Type I error).
* False Negative (FN): The number of positive instances incorrectly predicted as negative (Type II error).

4.1 Accuracy:

* Accuracy measures the overall correctness of the model's predictions.
* Formula: (TP + TN) / (TP + TN + FP + FN)

4.2 Precision:

* Precision measures the accuracy of positive predictions made by the model.
* Formula: TP / (TP + FP)

4.3 Recall:

* Recall measures the ability of the model to correctly identify positive instances.
* Formula: TP / (TP + FN)

4.4 F1-score:

* F1 score is the harmonic mean of precision and recall. It balances precision and recall, especially when dealing with imbalanced datasets.
* Formula: 2 \* (Precision \* Recall) / (Precision + Recall)

4.5 ROC curve:

The ROC curve (Receiver Operating Characteristic) is an evaluation method for binary classification models in machine learning. It visually represents the trade-off between a model's ability to correctly classify positive instances (True Positive Rate) and its tendency to classify negative instances as positive (False Positive Rate) at various threshold levels. The area under the ROC curve (AUC-ROC) quantifies overall model performance, with higher values indicating better discrimination ability. It's a valuable tool for comparing models and selecting thresholds that balance precision and recall based on specific priorities.

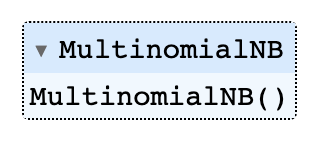
**4. Proposed system architecture and its implementation:**

This section mentioned the details about the models architecture and also the detail of the dataset.

We have used three models – Multinomial Naïve Bayes, Random Forest and deep neural network.

* 1. Naïve Bayes

We have used the library Scikit-learns MultinomialNB class to implement the Naïve Bayes model, where parameters value are set to default ones.

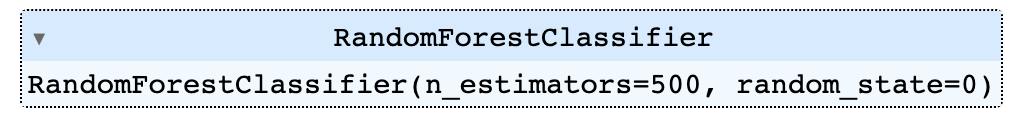


{'alpha': 1.0, 'class\_prior': None, 'fit\_prior': True, 'force\_alpha': 'warn'}

These are the default parameters.

* 1. Random Forest

We have used the library Scikit-learns RandomForestClassifier to implement the random forest model



We have set n\_estimators to 500 which means 500 Decision Trees are used, more its value more the model get robust and accurate.

{'bootstrap': True, 'ccp\_alpha': 0.0, 'class\_weight': None, 'criterion': 'gini', 'max\_depth': None, 'max\_features': 'sqrt', max\_leaf\_nodes': None, 'max\_samples': None, 'min\_impurity\_decrease': 0.0, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'min\_weight\_fraction\_leaf': 0.0, 'n\_estimators': 500, 'n\_jobs': None, 'oob\_score': False, ‘random\_state': 0, ‘verbose': 0, 'warm\_start': False}

* 1. Bidirectional LSTM (RNN)

In this we have used Sequential Model from Keras library. We have used different layers such as Bidirectional LSTM, Embedding layer, Dense layer and Dropout layer. We have used ‘ReLu’ activation function in hidden layers and ‘sigmoid’ activation function for output layer. Let’s view the model below –

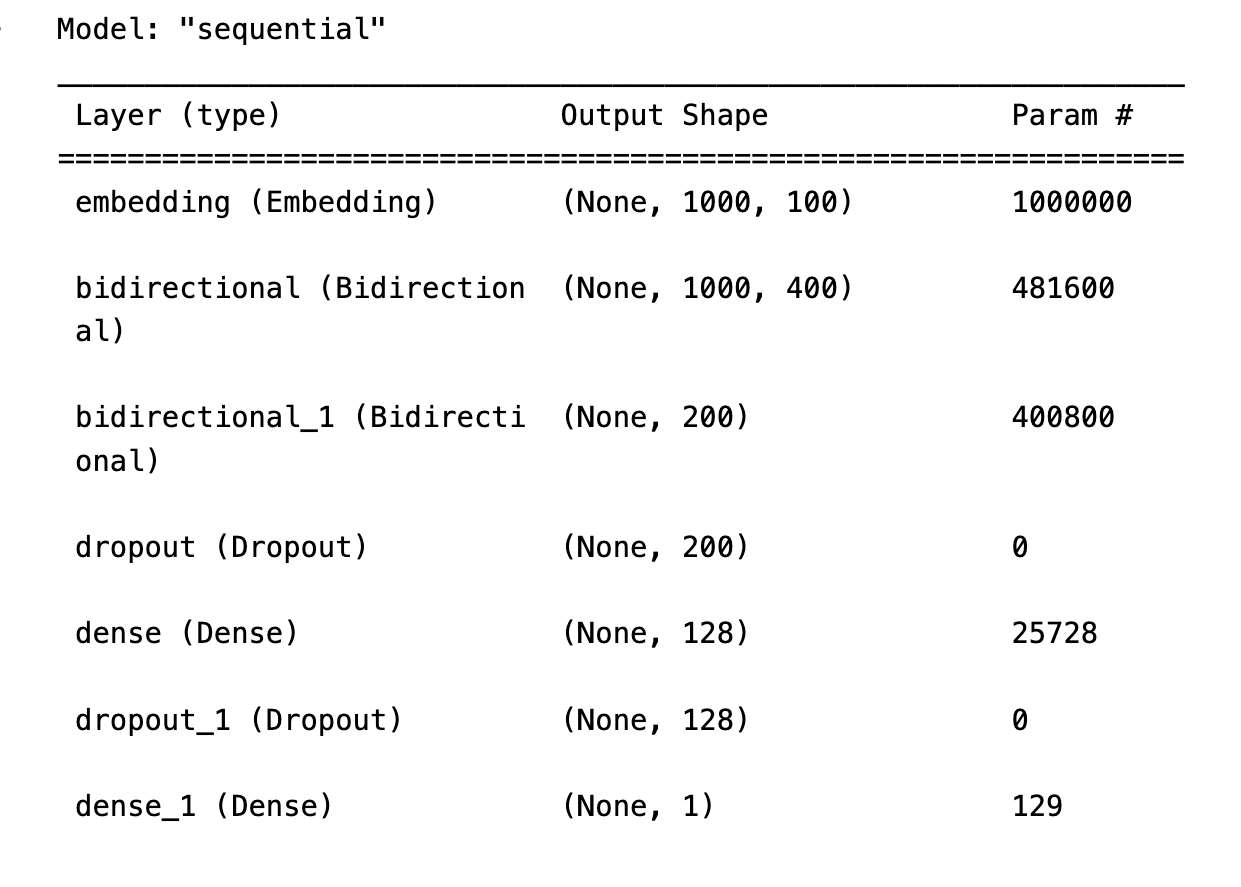


Fig 2: Layers details of Deep learning model

We have compiled the model with loss function as ‘binary cross entropy’, optimizer as ‘Adam’ and metrics as ‘accuracy.

And we fit the model with batch size = 50, epochs = 3 and validation data In this model we have taken input of size 1000 words.

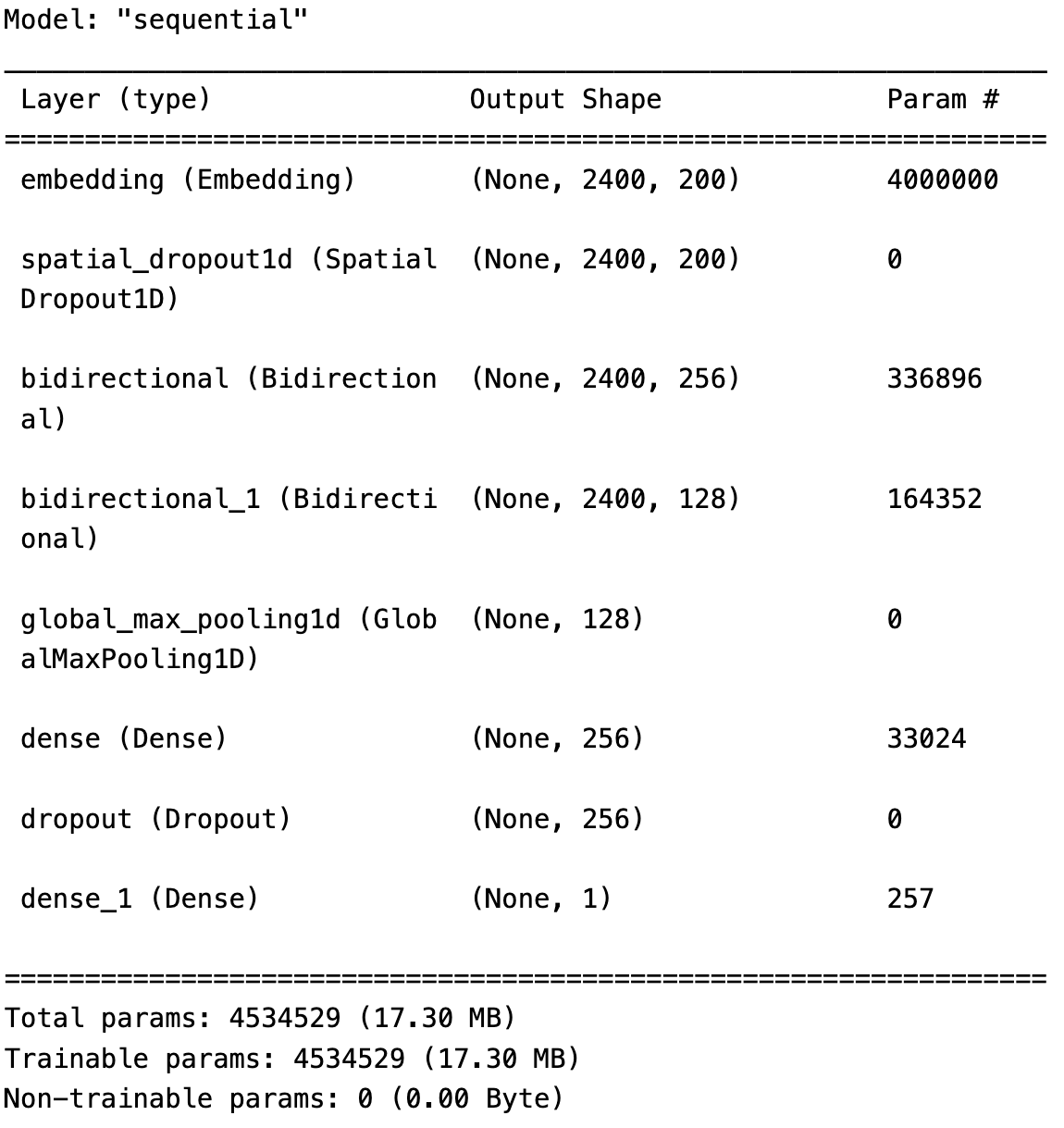
After this we have check the model performance by increasing more layers. In this model we have added SpatialDropout1D and GlobalMaxPooling1D layer. The GlobalMaxPooling1D layer is particularly useful when you want to maintain the most important features in your data and reduce the dimensionality, which can help prevent overfitting and improve model generalization. By adding SpatialDropout1D, you introduce a regularization technique that can improve the model's generalization and accuracy. The dropout rate (0.2) means that, during training, approximately 20% of the input elements along the time dimension will be set to zero during each forward and backward pass. This helps the model learn a more robust representation of your sequential data. The new model structure is shown below:   


Fig 3: After adding more layers

In this model we have tested the performance of the model on different input length of 800, 1600 and 2400 words.

We keep the compile parameter same and also the fit parameter same.

4.BERT model:

We have used BERT model from Hugging Face transformers library.

model\_name = 'bert-base-uncased'

We used AutoTokenizer to tokenize the data.

About Dataset:

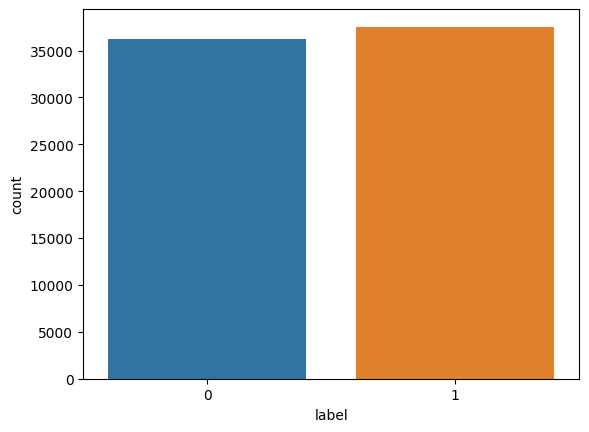
We have downloaded the fake news datasets from online sources and merge them in a single csv file. The dataset contains two columns one is ‘text’ and other one is ‘label’. The text column contains all the news of both Hindi and English language and the label column contains 0 for fake news and 1 for true news. The dataset contains total of 73775 news from which 36269 fake news and 37506 real or true news.

Fig 4: The plot showing number of true and fake news

**5.** **Experimental Results**

This section contains the results of our models such as accuracy, precision, F1 score, ROC curve, recall and confusion matrix.

5.1 Multinomial Naïve Bayes results

Accuracy: 0.9071162317858353

Precision: 0.9351752695081031

Recall: 0.8772435038842754

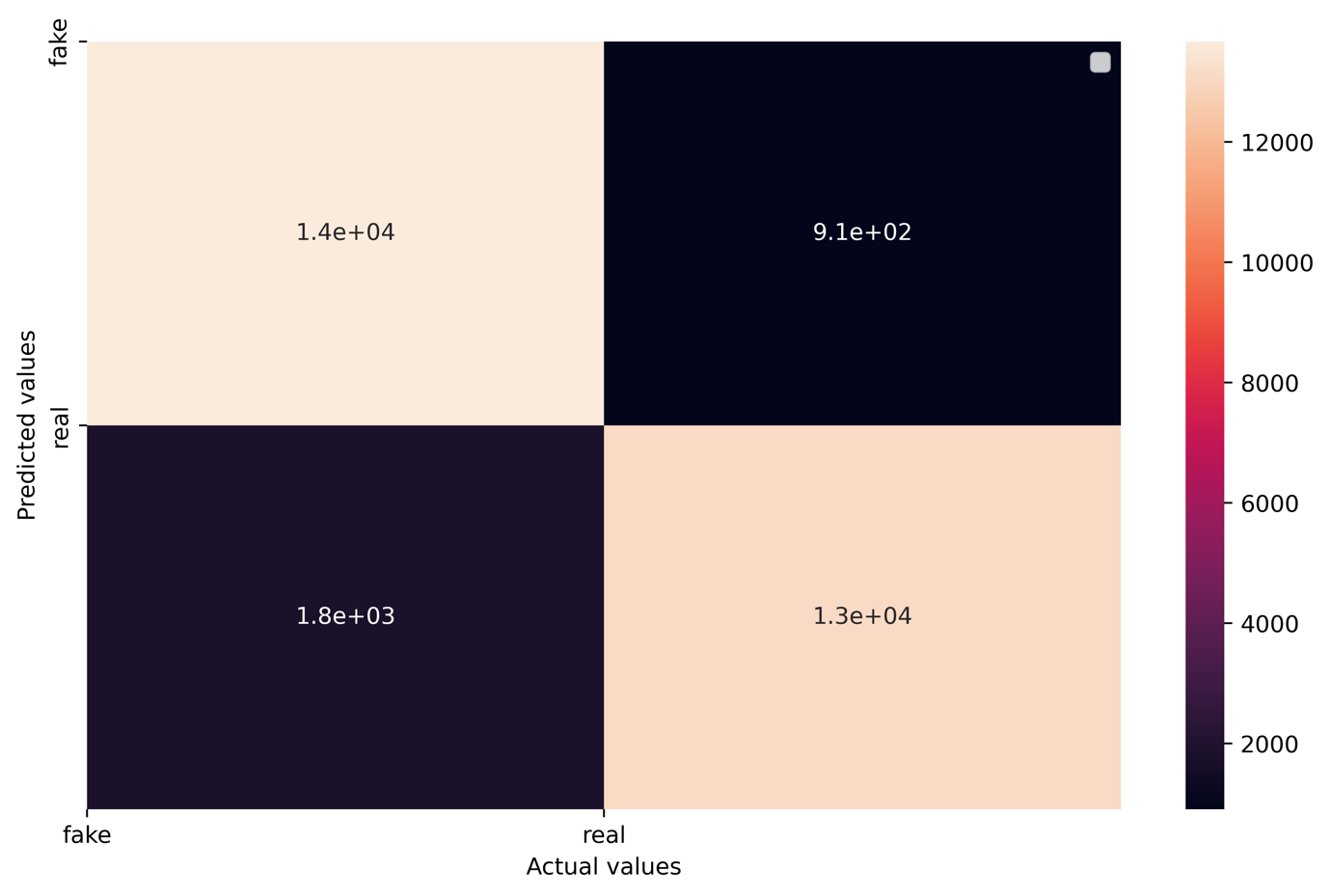
F1 Score: 0.905283527419745

Fig 5: Confusion matrix of Naïve Bayes

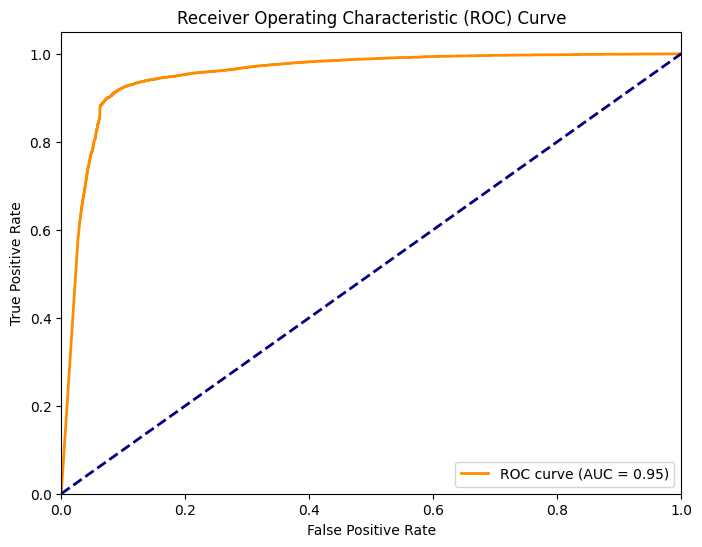


Fig 6: ROC curve of Naive Bayes

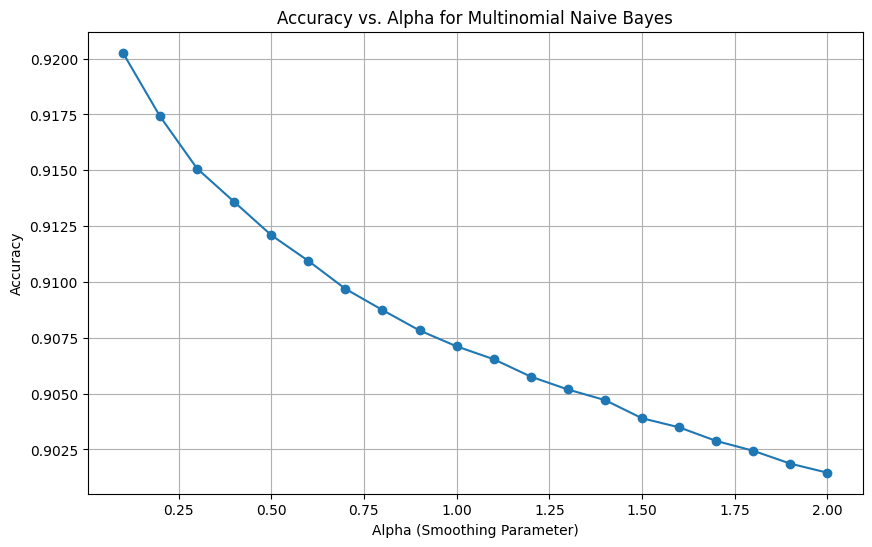


Fig 7. Accuracy curve of Naïve Bayes on different Alpha(smoothing parameter)

5.2 Random Forest

Accuracy: 0.9399525584547611

Precision: 0.9438149197355996

Recall: 0.9371149209750871

F1 Score: 0.940452987431951

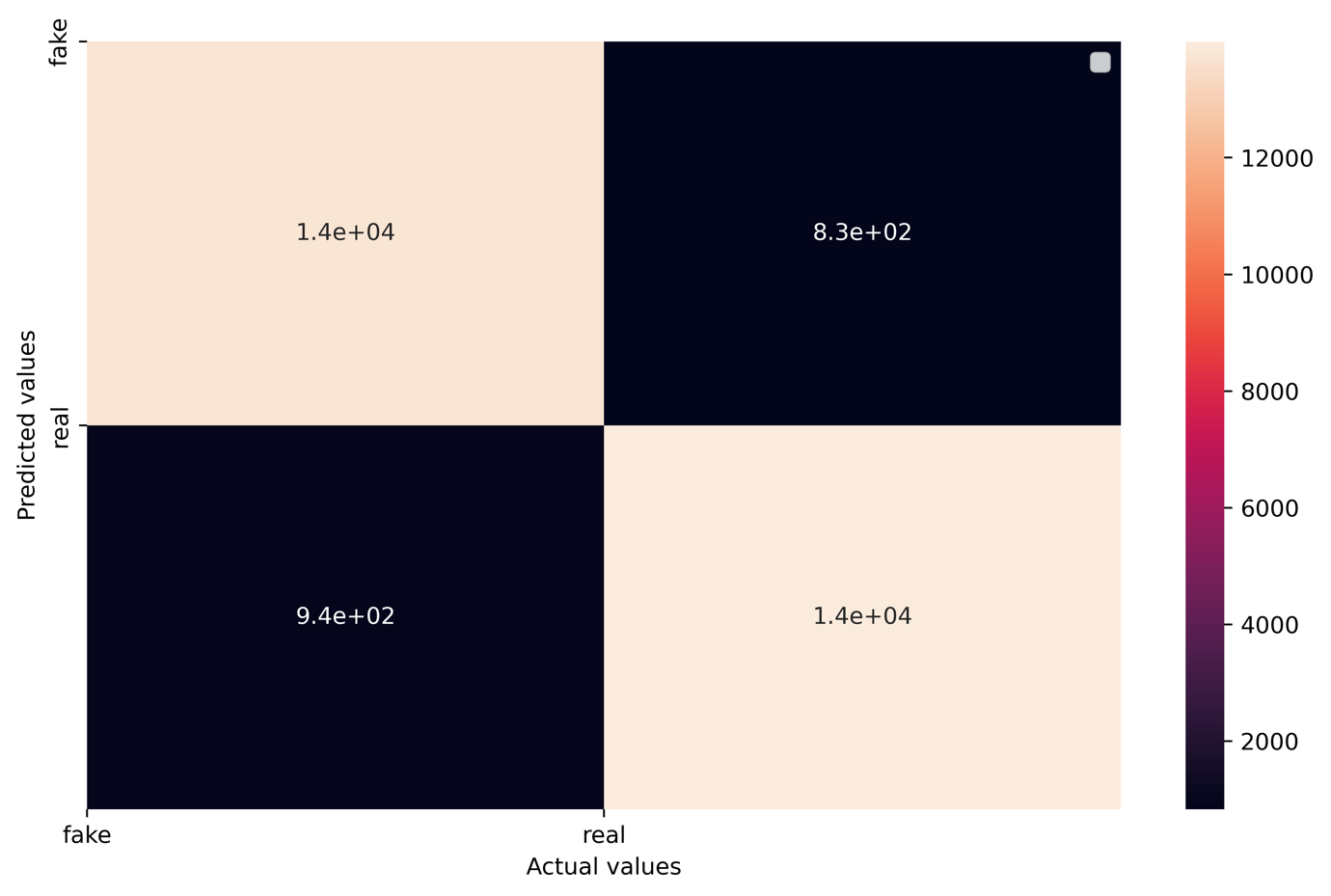


Fig 8: Confusion matrix of Random Forest

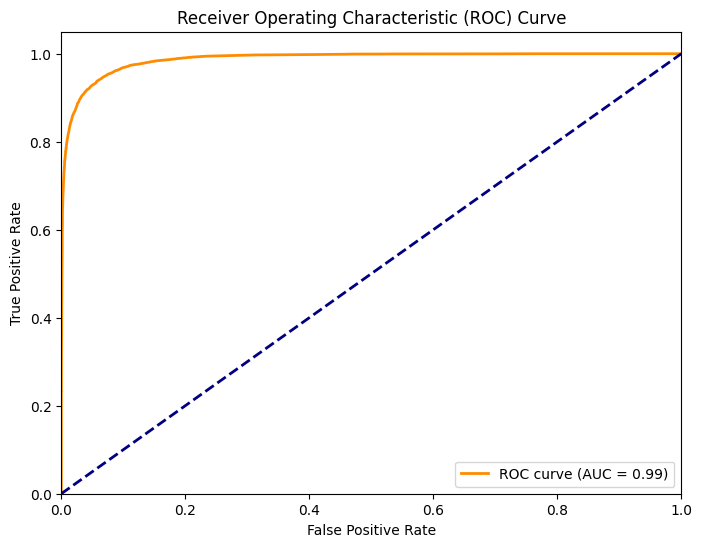


Fig 9: ROC graph of Random Forest

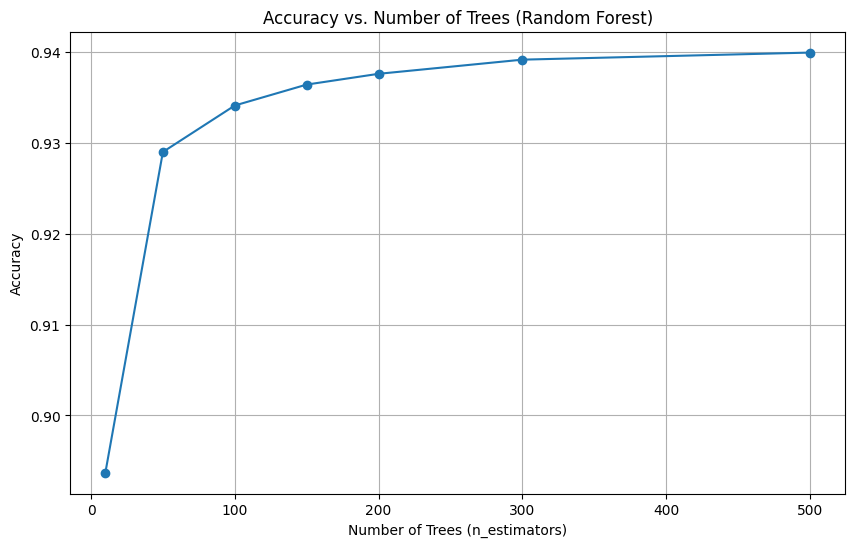


Fig 10: Accuracy curve of Random Forest on different number of Decision trees(n\_estimators)

5.3 Bidirectional LSTM

This is the accuracy of the first model on 1000 words input length:

Accuracy: 0.9486953575059301

Precision: 0.964934164934165

Recall: 0.9324939726761318

F1 Score: 0.9484367549894422

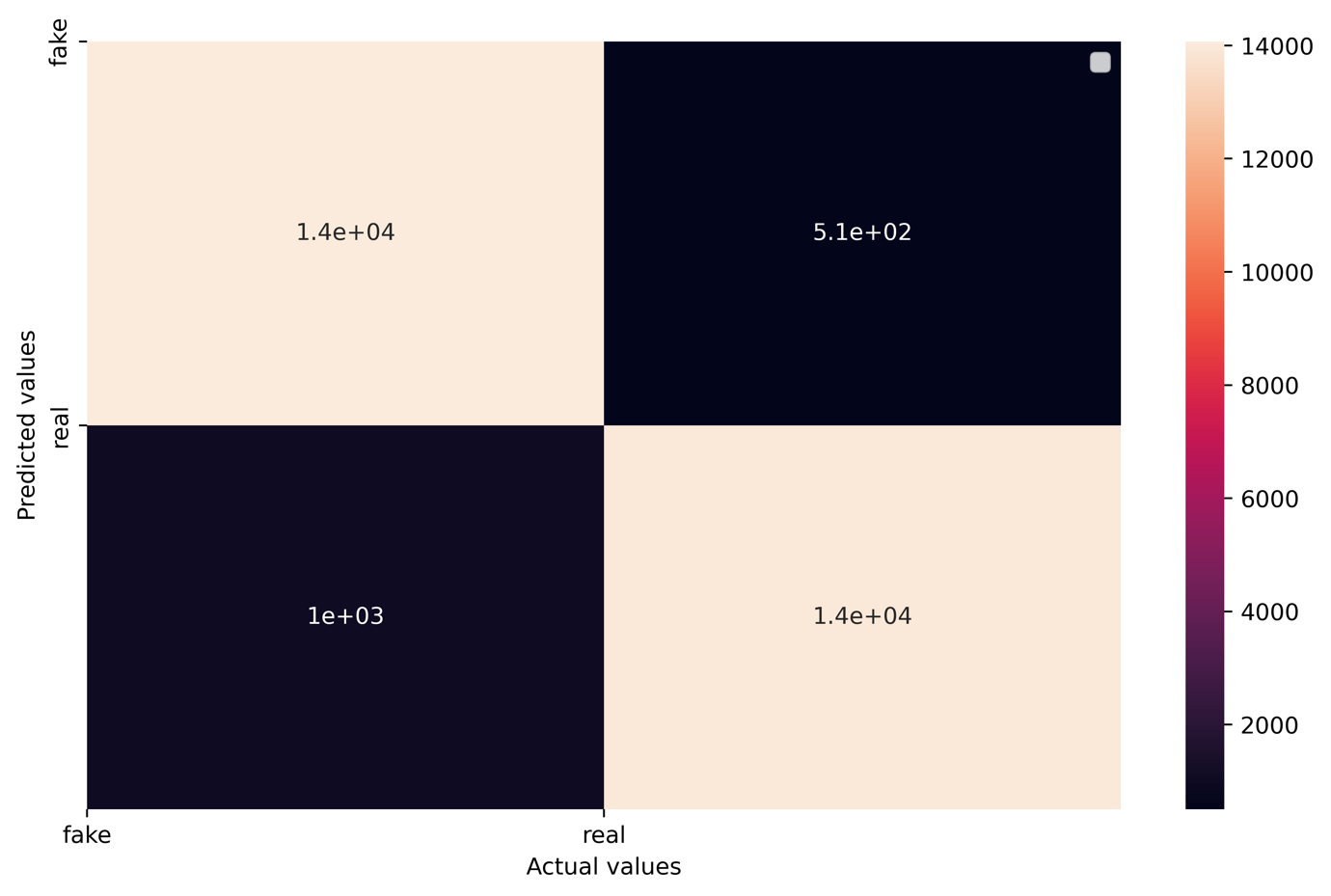


Fig 11: LSTM confusion matrix

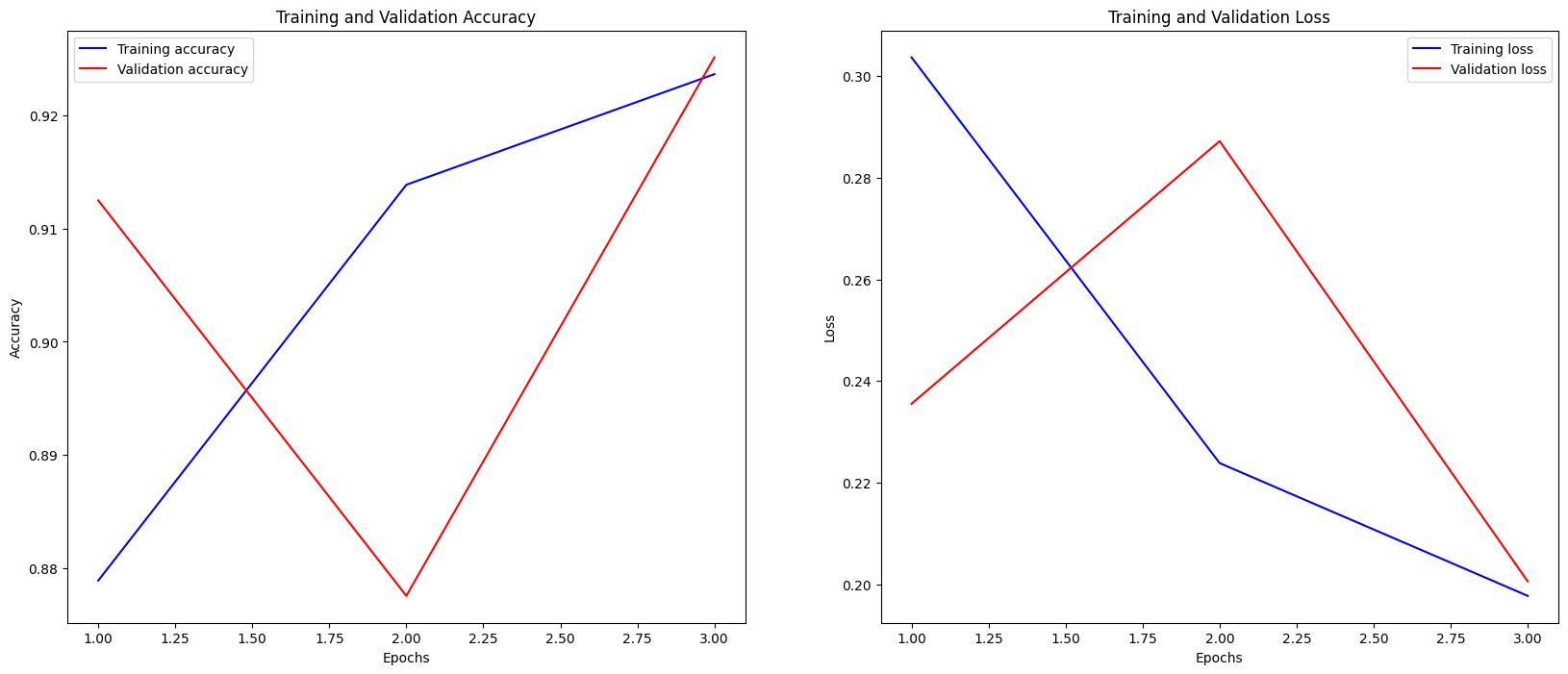


Fig 12: Accuracy and Loss curve of LSTM model based on number of epochs

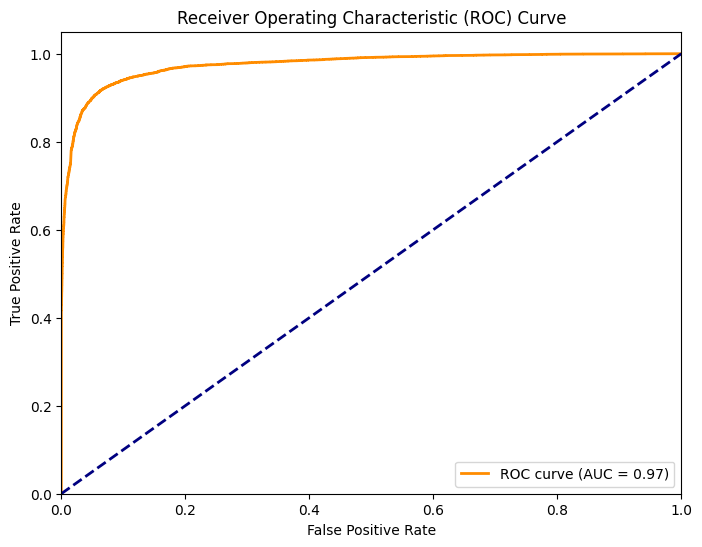


Fig 13: ROC curve of LSTM model

* Now the second model on input length 800 words:

Accuracy: 0.9812053853799584

Precision: 0.9814814814814815

Recall: 0.9811286843997125

F1 Score: 0.98130505123135

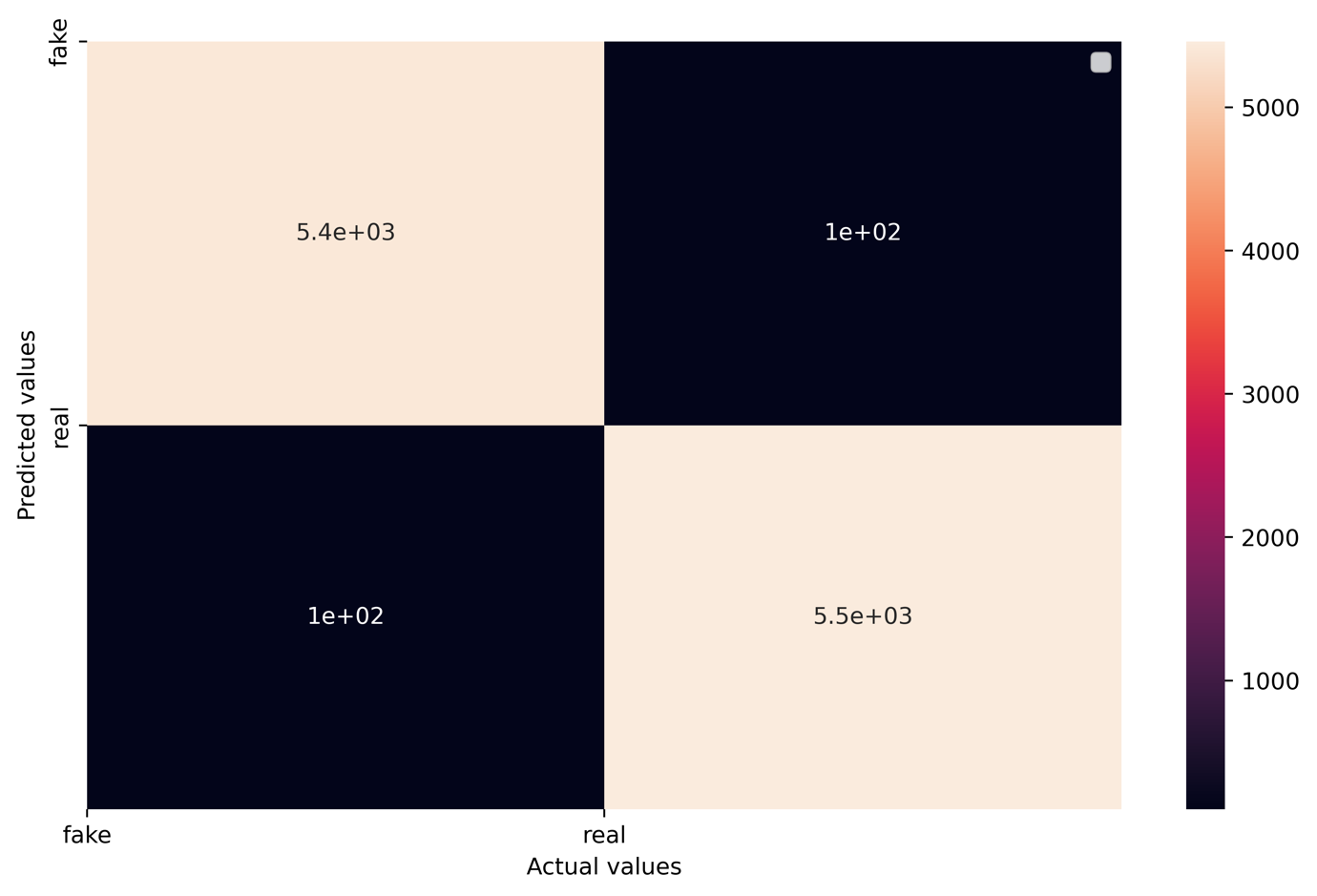


Fig 14: Confusion matrix of 2nd LSTM model for 800 words input length

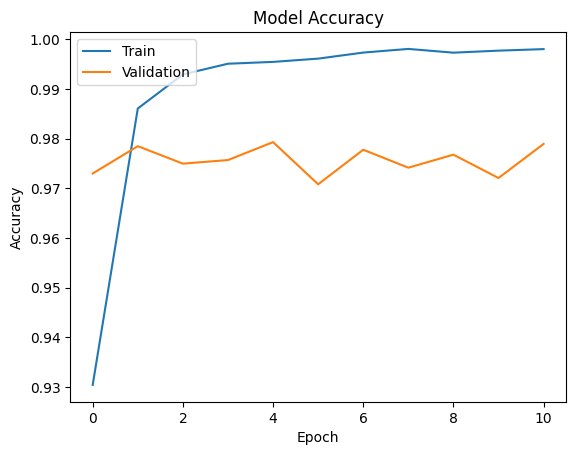


Fig 15: Accuracy curve of 2nd LSTM model with 800 words input length

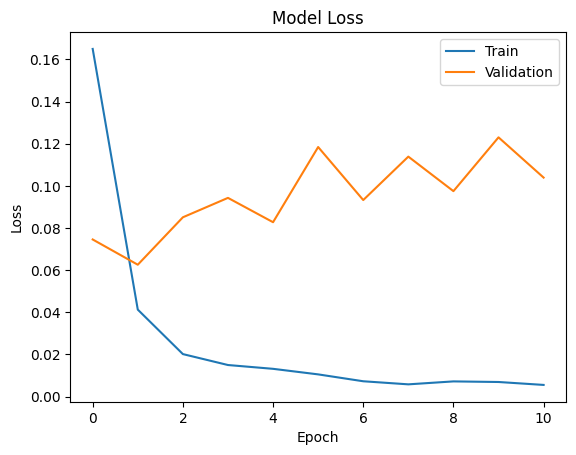


Fig 16: Loss curve of 2nd LSTM model for 800 words input length

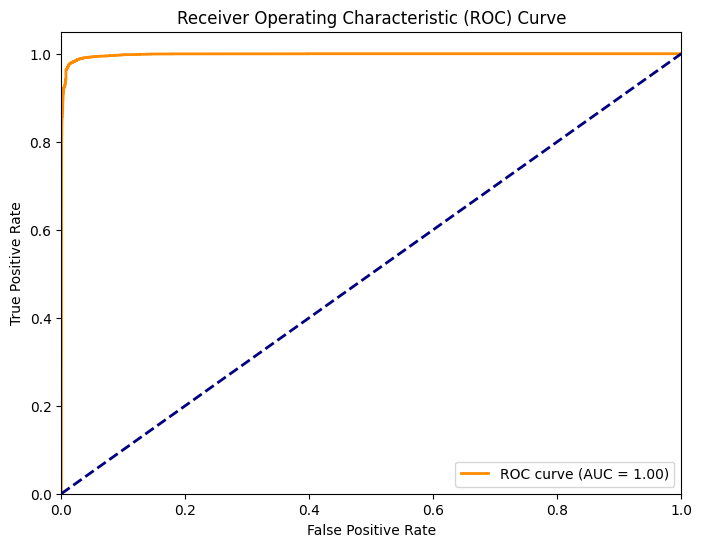


Fig 17: ROC curve of 2nd LSTM model for input length 800 words

* Now the second model on input length 1600 words:

Accuracy: 0.9838257883798681

Precision: 0.9834799784521459

Recall: 0.9843637670740475

F1 Score: 0.9839216743016258

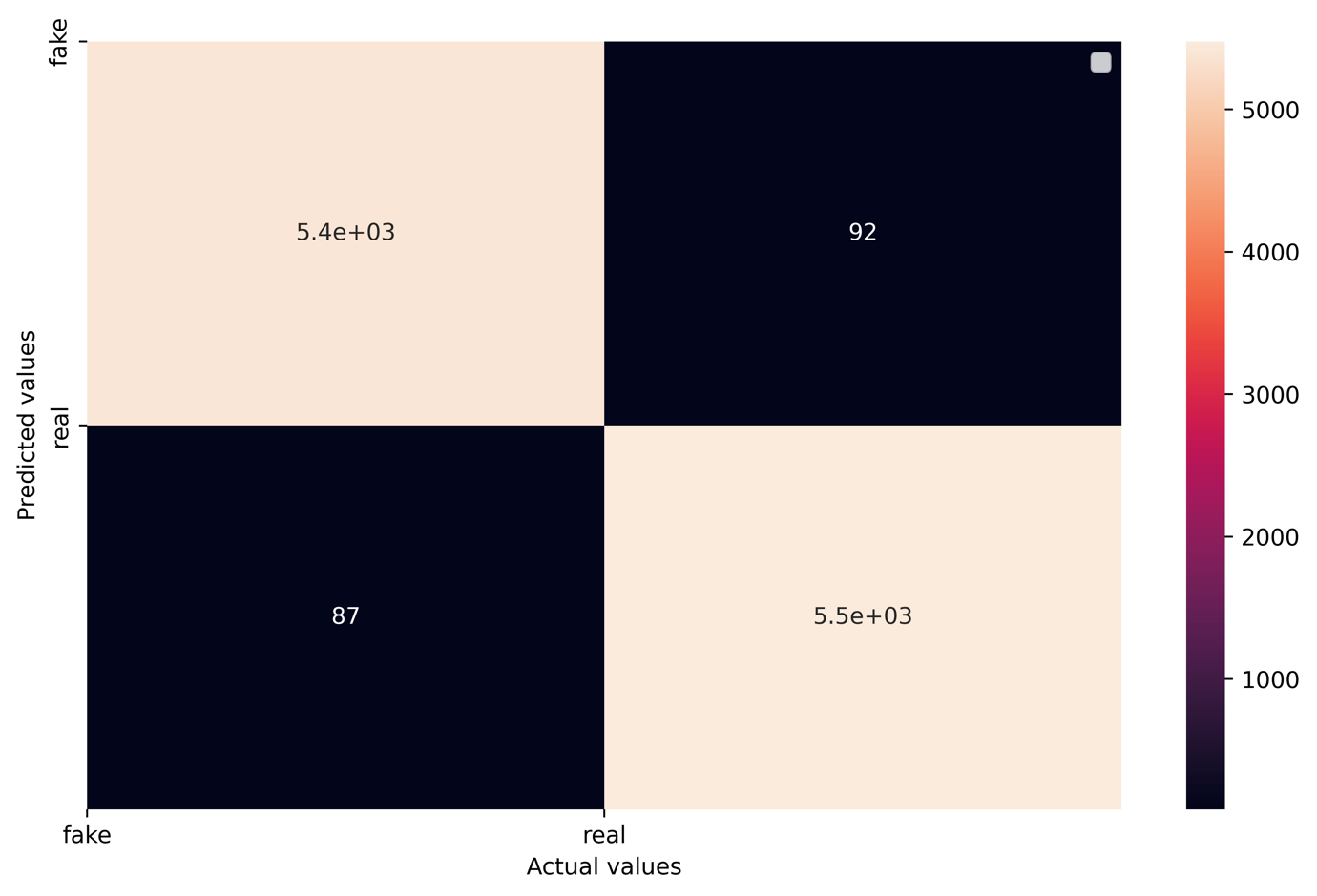


Fig 18: Confusion matrix of 2nd LSTM model for 1600 words input length

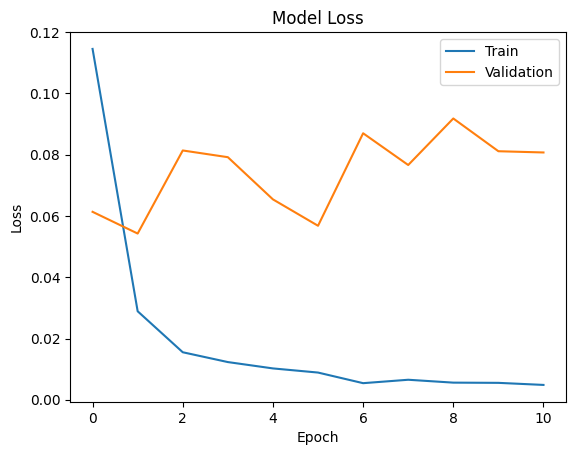
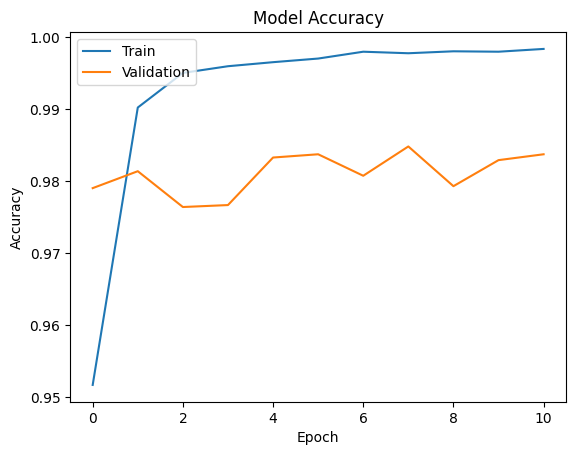


Fig 19: Accuracy and Loss curve of 2nd LSTM model for 1600 words input length

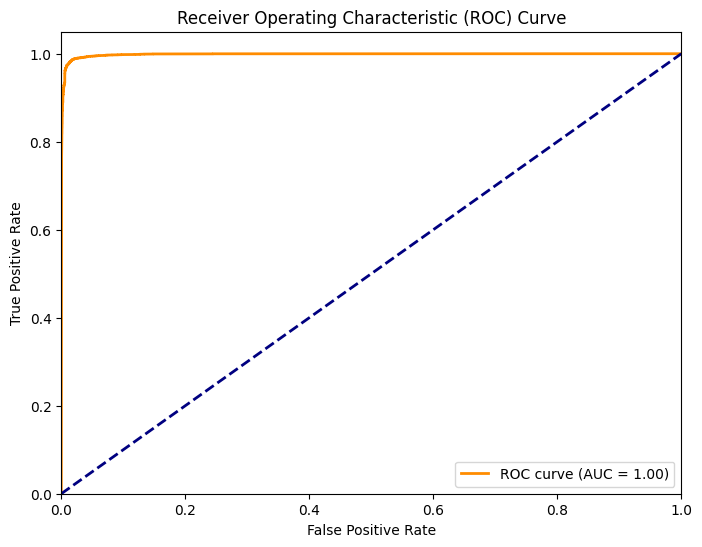


Fig 20: ROC curve of 2nd LSTM model for 1600 words input length

* Now the second model on input length 2400 words:

Accuracy: 0.9811150266558236

Precision: 0.9758308157099698

Recall: 0.9868799424874192

F1 Score: 0.9813242784380306

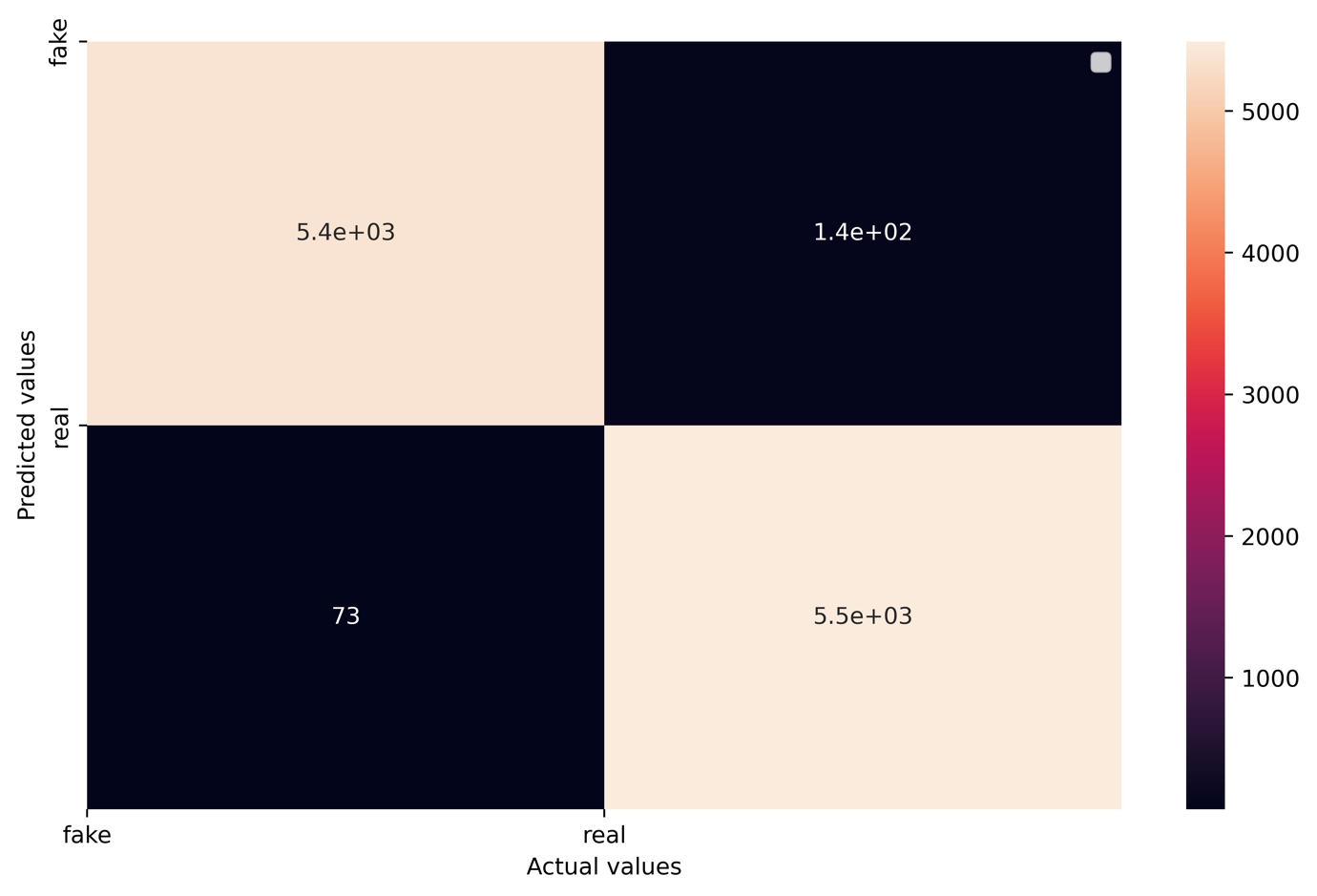
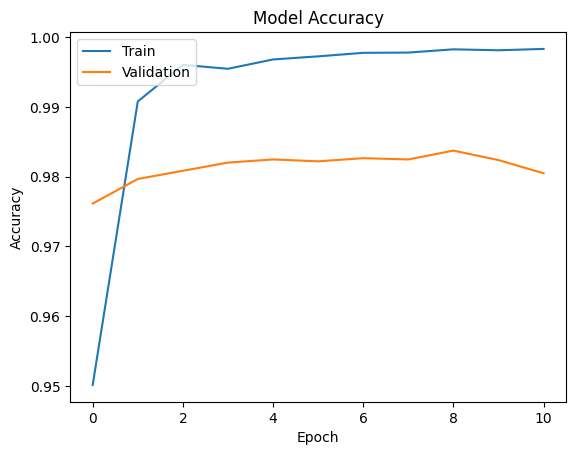


Fig 21: Confusion matrix of 2nd LSTM model for 2400 words input length



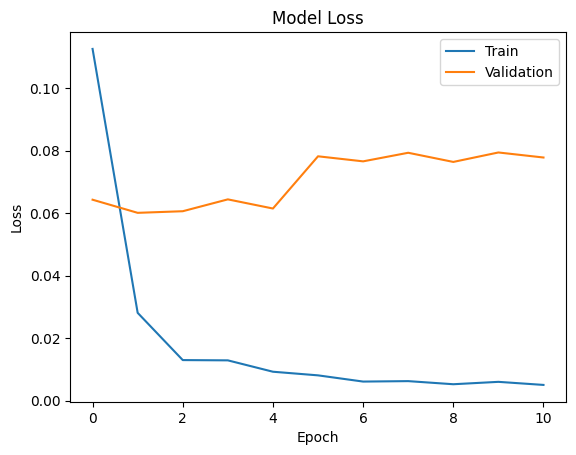


Fig 22: Accuracy and Loss curve of 2nd LSTM model for 800 words input length

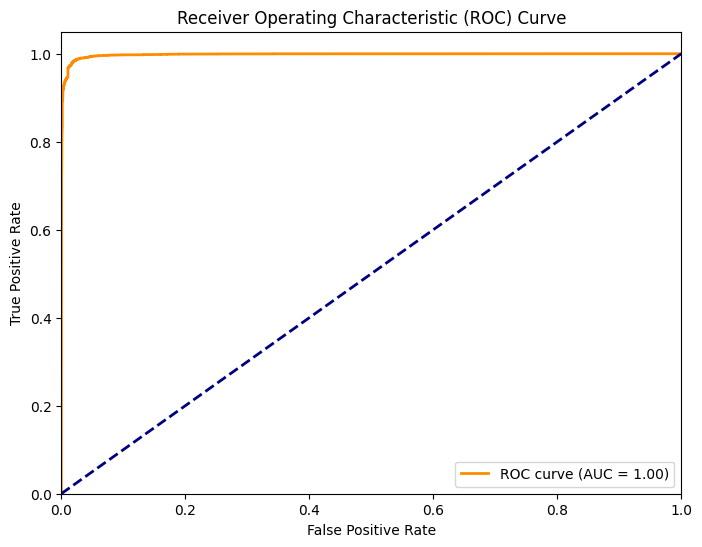


Fig 23: ROC curve of 2nd LSTM model for 2400 words input length

I have applied score fusion techniques on these three input length models:

Average fusion : 0.9820487334718834

Max fusion: 0.9838257883798681

Weighted fusion: 0.9822264389626819

Sum of product fusion: 2.8932567742569066

5.4 BERT model :

Accuracy: 0.989247311827957

Precision: 0.9915147138472649

Recall: 0.98705966930266

F1 Score: 0.9892821759884717

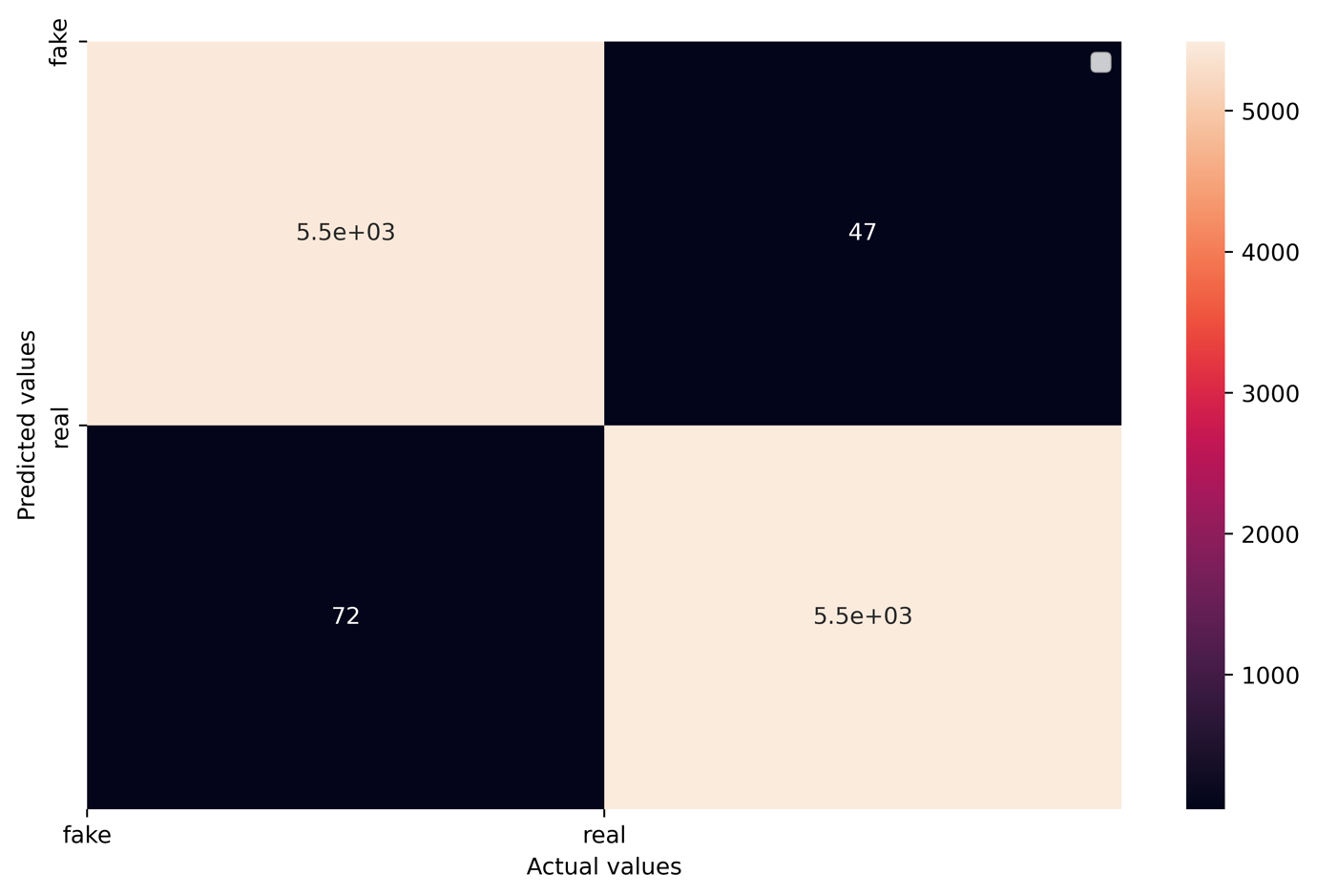


Fig 24: Confusion matrix of BERT model

5.4 Model comparison

**BERT Model**: The BERT model exhibited the highest accuracy among all models, with an impressive accuracy of 98.9%. BERT, a state-of-the-art pre-trained language model, demonstrated its effectiveness in this task.

**2nd Model with Input Length 2400**: The second model, when fed with sequences of length 2400, achieved an accuracy of 98.1%. It performed exceptionally well, and the longer input sequences seemed to be beneficial for this specific task.

**2nd Model with Input Length 1600**: When using shorter input sequences of length 1600, the same model still maintained high accuracy at 98.4%. This suggests that it can handle both long and shorter text inputs effectively.

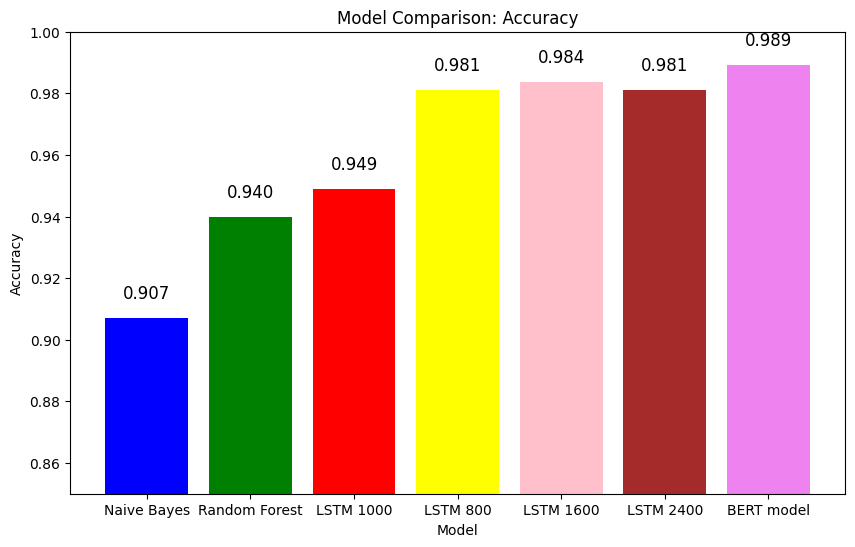
**2nd Model with Input Length 800**: The model's performance remained consistent with input sequences of length 800, maintaining an accuracy of 98.1%. This flexibility could be advantageous for various use cases.

**1st Model with Input Length 1000**: The first model, designed for input sequences of length 1000, achieved an accuracy of 94.9%. Although it didn't outperform BERT or the second model, it still demonstrated respectable performance.

**Random Forest**: The Random Forest model exhibited an accuracy of 94.0%. While it's a traditional machine learning model, it performed well in this context.

**Naive Bayes**: The Naive Bayes model achieved an accuracy of 90.7%. Despite being a simple and fast algorithm, it provided a baseline performance level.

In summary, the BERT model outperformed all other models with the highest accuracy. The second model showcased remarkable flexibility by maintaining high accuracy across different input sequence lengths. The Random Forest and Naive Bayes models performed well but couldn't match the accuracy achieved by the deep learning models. These findings suggest that, depending on the use case and available resources, different models can be selected to achieve the desired level of accuracy.

Fig 25: Accuracy of all models

6. **Conclusions and Future works**

In this study, we conducted an extensive comparative analysis of three machine learning models: Naive Bayes, Random Forest, and Long Short-Term Memory (LSTM) Deep Learning, for the task of Fake news classification. Each model demonstrated distinct strengths and weaknesses, offering valuable insights for both research and practical applications.

Future research directions include exploring model fusion techniques, advanced feature engineering, hyperparameter tuning, interpretable models, and the applicability of transfer learning. Evaluating real-world deployment considerations such as computational efficiency and scalability is also essential for practical applications.

7. **References:**

[1] Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). "Fake News Detection on Social Media: A Data Mining Perspective." ACM SIGKDD Explorations Newsletter, 19(1), 22-36.

[2] Ruchansky, N., Seo, S., & Liu, Y. (2017). "CSI: A Hybrid Deep Model for Fake News Detection." arXiv preprint arXiv:1703.06959.

[3] Volkova, S., Shaffer, K., Jang, J. Y., Hodas, N. O., & Bell, E. (2017). "Separating Fact from Fiction: An Examination of Deceptive Self-Presentation in Online Dating Profiles." In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (Vol. 1, pp. 285-296).

[4] Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). "Fake News Detection on Social Media: A Data Mining Perspective." ACM SIGKDD Explorations Newsletter, 19(1), 22-36. ↩

[5] Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). "Fake News Detection on Social Media: A Data Mining Perspective." ACM SIGKDD Explorations Newsletter, 19(1), 22-36.

[6] Ruchansky, N., Seo, S., & Liu, Y. (2017). "CSI: A Hybrid Deep Model for Fake News Detection." arXiv preprint arXiv:1703.06959.

[7] Volkova, S., Shaffer, K., Jang, J. Y., Hodas, N. O., & Bell, E. (2017). "Separating Fact from Fiction: An Examination of Deceptive Self-Presentation in Online Dating Profiles." In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (Vol. 1, pp. 285-296).

[8] Wu, Y., Shen, H., Du, B., Chen, L., & Cheng, J. (2021). "Multimodal Fake News Detection with Visual and Textual Information." Information Sciences, 581, 296-312.

[9] Friggeri, A., Adamic, L. A., Eckles, D., & King, J. (2014). "Rumor Cascades." Proceedings of the Eighth International Conference on Weblogs and Social Media, 101-110.