MAIM

Fake and Real News

Name: Abdelrahman Saied Abdelrahman

Abstract

This project explores the task of text classification for Fake and Real News using both classical machine learning models and transformer-based architectures. The objective was to preprocess the dataset, train multiple models, and evaluate their performance to determine the effectiveness of modern deep learning methods compared to traditional approaches. The results indicate that transformer-based models outperform classical methods in terms of accuracy and more ML metrics.

preprocessing → classical ML → deep sequence models → transformers → final deployment

---

Introduction

Text classification is a fundamental NLP, with applications ranging from sentiment analysis to fake news detection. Traditional machine learning models, such as Naive Bayes, Logistic Regression, and Support Vector Machines, have historically been applied to this task using effective features.

Then, using sequential models such as RNN, LSTM, and GRU.
Then, using advancements in transformer architectures, such as Transformer-Based DistilBERT, has revolutionized NLP by enabling models to catch deeper contextual information.
This project aims to compare classical ML approaches, sequential models, and transformer fine-tuning to identify the strengths and limitations of each.

Dataset description

dataset link:
https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset

The dataset is separated into two files:

1. Fake.csv: News articles identified as not genuine (23502 fake news articles)
2. True.csv: News articles identified as authentic (21417 true news articles)

Total Samples: 44,919 articles

Dataset columns:
Title: title of news article
Text: body text of news article
Subject: Subject of news article
Date: publish date of news article

Purpose:
This dataset is designed for binary text classification tasks (fake vs real news) and is widely used in NLP and Machine Learning projects.

Methodology and Results

1. Preprocessing:

First: Importing many libraries to read, plot, export, split and pre-processing the data

reading data and mapping the subject into [world ,politics] rather than old mapping.

True_df subject: politics 11272 world 10145
Fake df subject: politics 12870 world 10611

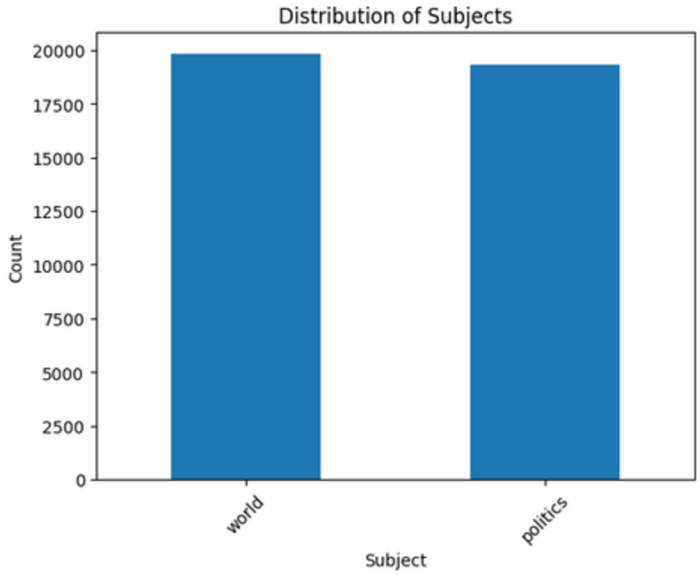
Then make a new column ‘class’, give the data label true or fake.

make a new df that concat the whole of data.

Then check null and duplicate values.
null value = 0 , duplicated values = 5777.
So We drop duplicates.

Then make many EDA for the data like info,describe and making some plots.

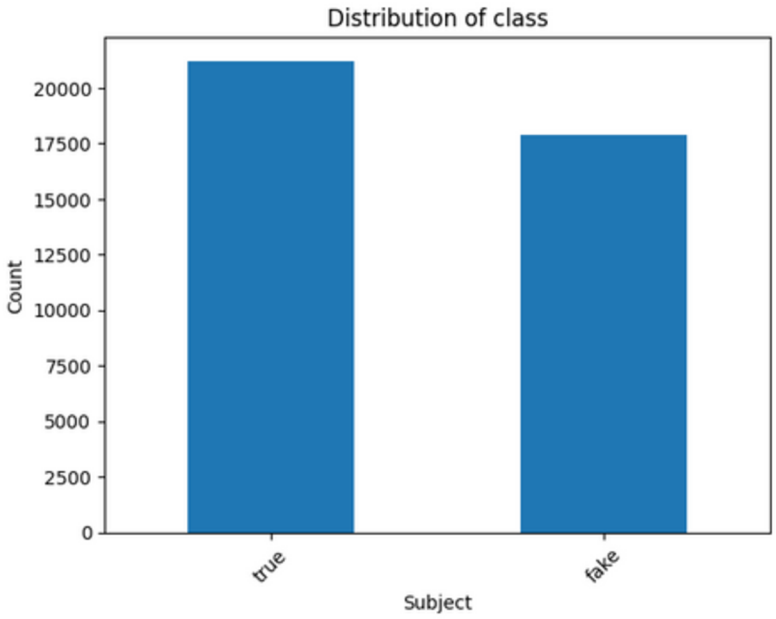
Distribution of Subjects:



Distribution of Subjects:

true: 21211
fake: 17910

true: 54.22 %
fake: 45.78 %



Then split the data into X, Y.
X = text of the data and Y = text's class.

Then split the data into train and test.
First, we export the data without preprocessing text to use it in transformer and fine-tuning part.

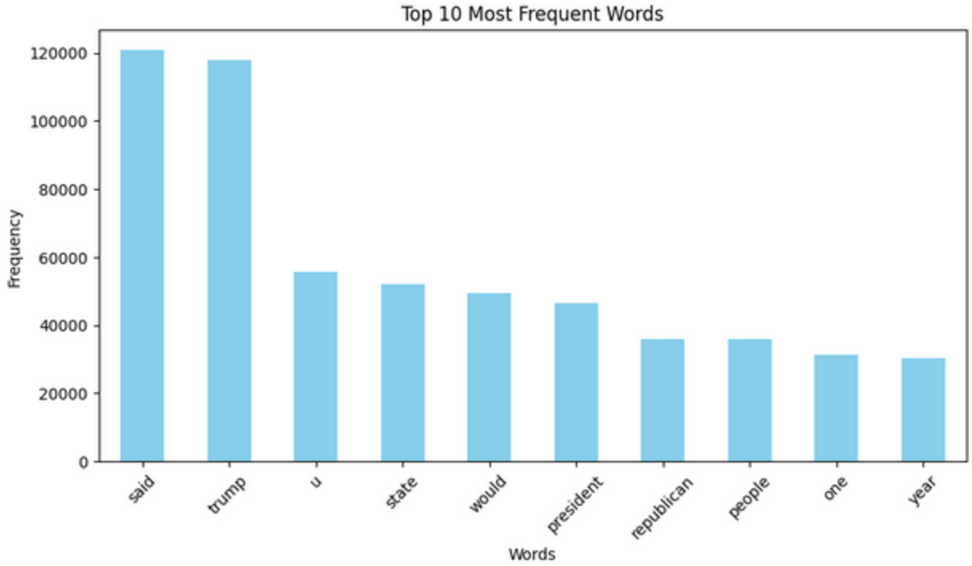
We export the data into a pkl file to use it again in another notebook.

Then, preprocessing the text using re to make the text without numbers or signs, word tokenizing to split text into tokens, lowercase to make all data in the same type, removing stop words to focus on important words, and lemmatizing to get the root of the word.

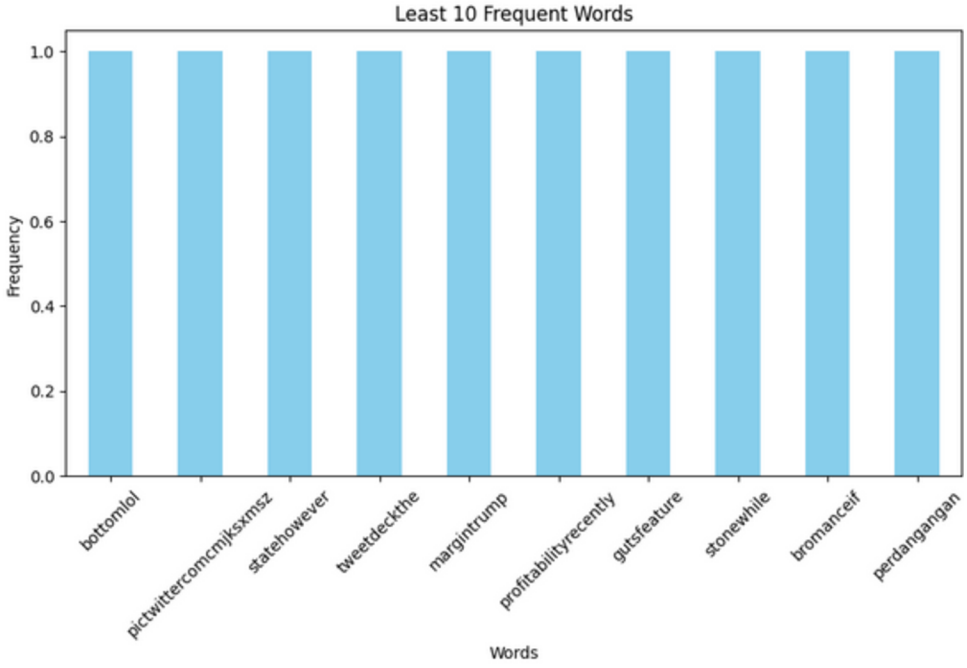
Then save df as a pkl file.

Then, use TF-IDF to preprocess and get many insights.

Top 10 Most Frequent Words:



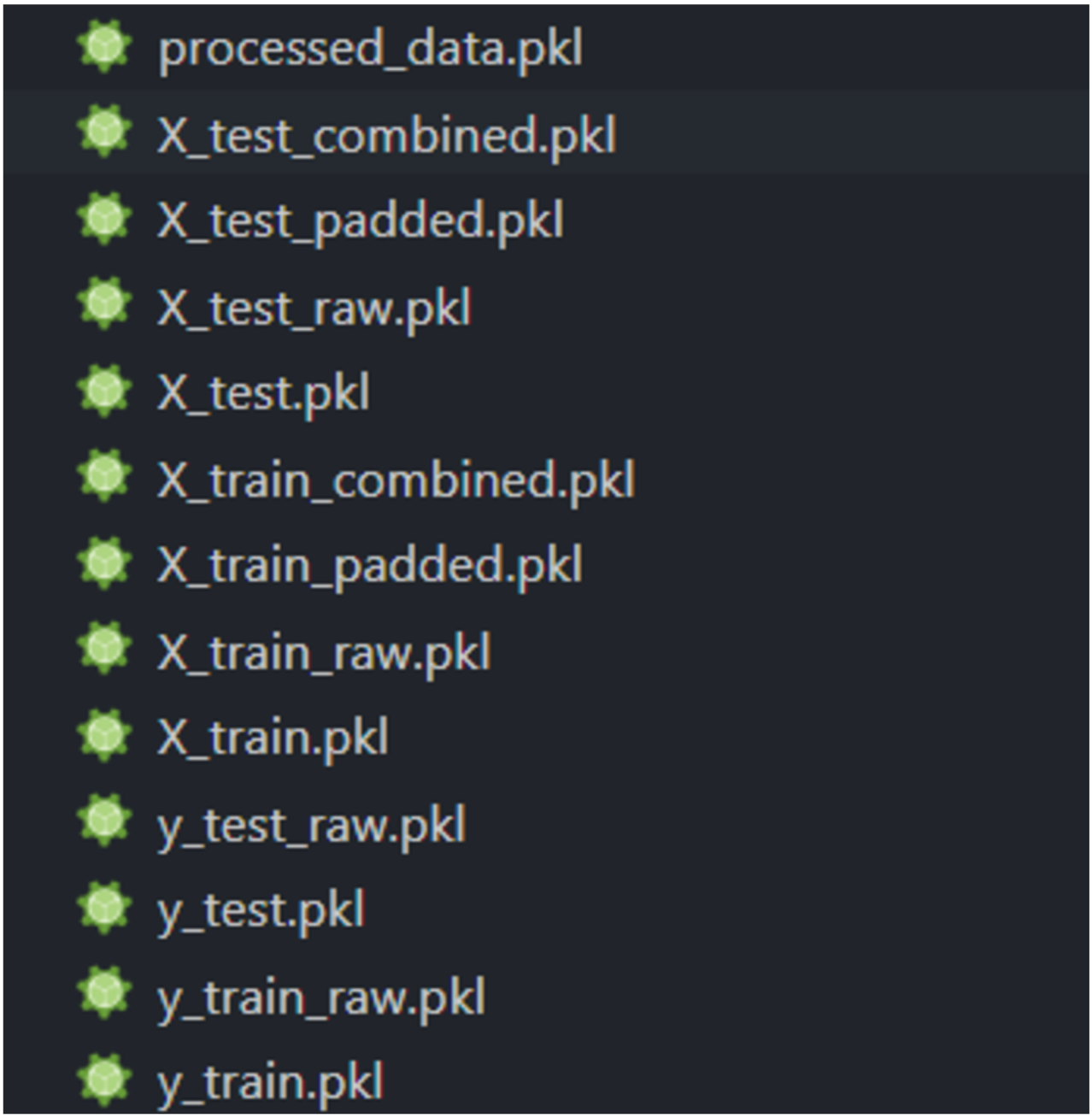
Least 10 Frequent Words:



Then, train the model on TF-IDF and BOW to use it in ML models.
After that, export X train and X test to use them.

Then, import tokitizer and pad_seq to use them in Sequential models.

Now we have many versions of X to use in ML models, Sequential models, Teransformer and Fine-Tuining



2. Classical ML models:

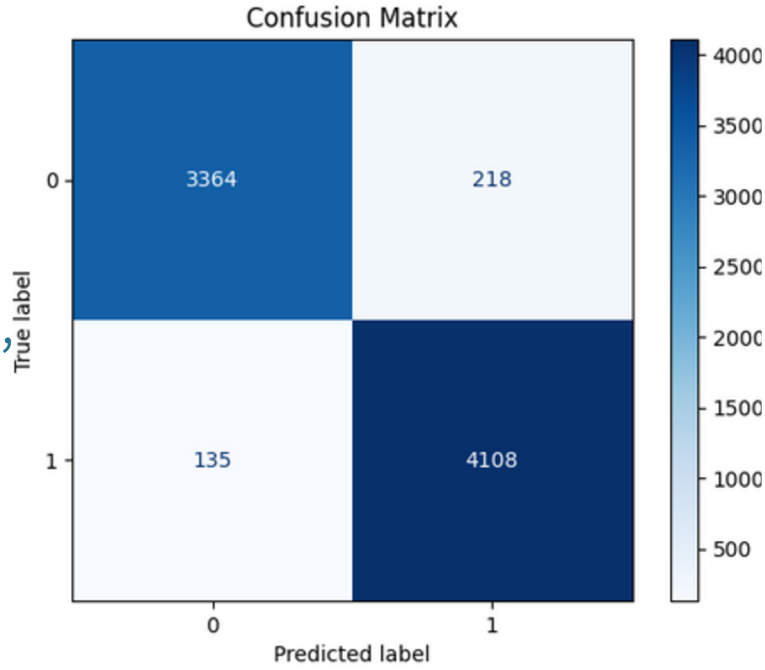
First: loading The right version of preprossed data.

import display confusion matrix and many libararies to make reports.

First model is Naive Bayse, which uses the GridSearchCV function to get the best accuracy. Then train the model on our data.

Best Naive Bayes params: {'clf__alpha': 0.1}
Best Naive Bayes score: 0.952622699386503

Accuracy: 0.95
Precision:0.94 (class0),0.97 (class 1)
Recall:0.96 (class 0), 0.95 (class 1)
F1-score: 0.96

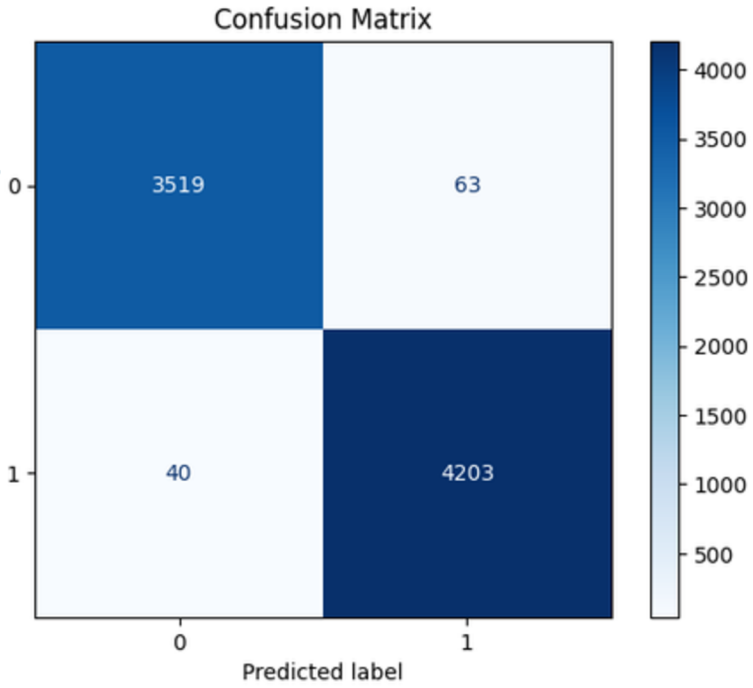


Class	Precision	Recall	F1-Score	Support
0 (Fake)	0.94	0.96	0.95	3499
1 (Real)	0.97	0.95	0.96	4326
Accuracy			**0.95**	7825
Macro Avg	0.95	0.96	0.95	7825
Weighted Avg	0.96	0.95	0.95	7825

The second model is logistic Regression, which uses the GridSearchCV function to get the best accuracy. Then train the model on our data.

Best Logistic Regression params: {'clf__C': 0.1, 'clf__penalty': 'l2', 'clf__solver': 'liblinear'}
Best Logistic Regression score: 0.9843110940695295

Accuracy: 0.99
Precision: 0.99 (class 0), 0.99 (class 1)
Recall: 0.99 (class 0), 0.99 (class 1)
F1-score: 0.99

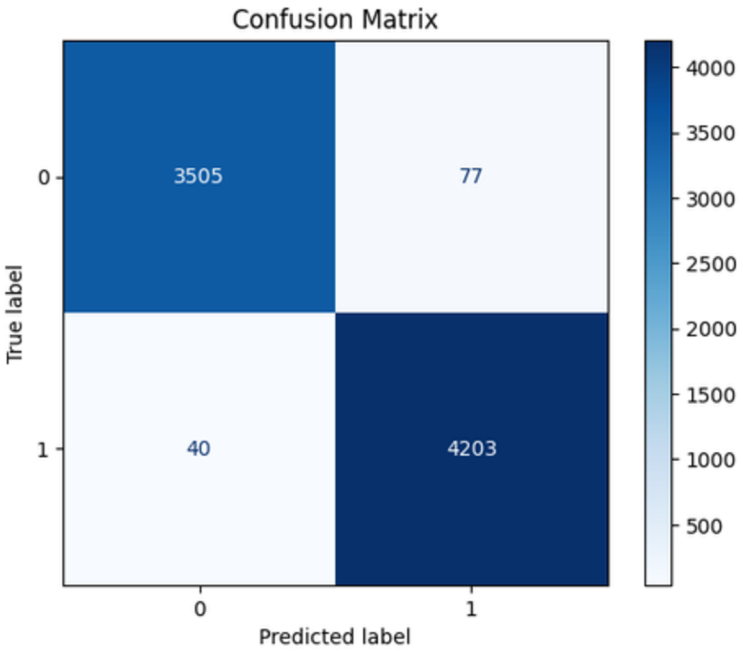


Class	Precision	Recall	F1-Score	Support
0 (Fake)	0.98	0.99	0.99	3559
1 (Real)	0.99	0.99	0.99	4266
Accuracy			**0.99**	7825
Macro Avg	0.99	0.99	0.99	7825
Weighted Avg	0.99	0.99	0.99	7825

The Third model is SVM, which uses the GridSearchCV function to get the best accuracy. Then train the model on our data.

Best SVM params: {'clf__kernel': 'rbf', 'clf__C': 10}
Best SVM score: 0.980604550102249 6

Accuracy: 0.99
Precision: 0.98 (class 0), 0.99 (class 1)
Recall: 0.99 (class 0), 0.98 (class 1)
F1-score: 0.98 – 0.99



Class	Precision	Recall	F1-Score	Support
0 (Fake)	0.98	0.99	0.98	3545
1 (Real)	0.99	0.98	0.99	4280
Accuracy			**0.99**	7825
Macro Avg	0.98	0.99	0.98	7825
Weighted Avg	0.99	0.99	0.99	7825

Conclusion :

Among the three models:

Naive Bayes is less accurate.

Logistic Regression outperformed others with the most consistent and balanced results (Accuracy = 0.99).

SVM also performed well, but still less than Logistic Regression.

So Best Model: Logistic Regression

2. Sequential models:

three models main parameters: max_words = 10000, embedding_dim = 128, input_length = 100,epochs = 10, batch_size = 64

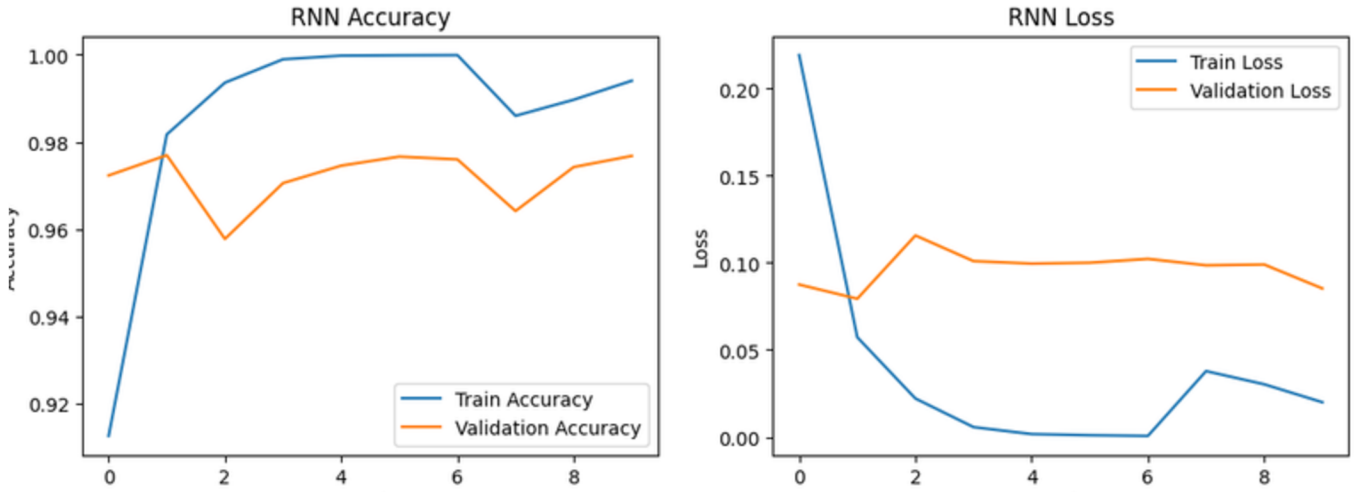
First model is RNN:

Training Results (10 Epochs):

- Epoch 1: Training Acc = 0.91, Val Acc = 0.97
- Epoch 2: Training Acc = 0.98, Val Acc = 0.98
- Epoch 3: Training Acc = 0.99, Val Acc = 0.96
- Epoch 4–7: Training Acc = 1.00, Val Acc = 0.97
- Epoch 8: Training Acc = 0.99, Val Acc = 0.96
- Epoch 9: Training Acc = 0.99, Val Acc = 0.97
- Epoch 10: Training Acc = 0.99, Val Acc = 0.98

Loss Results (10 Epochs)

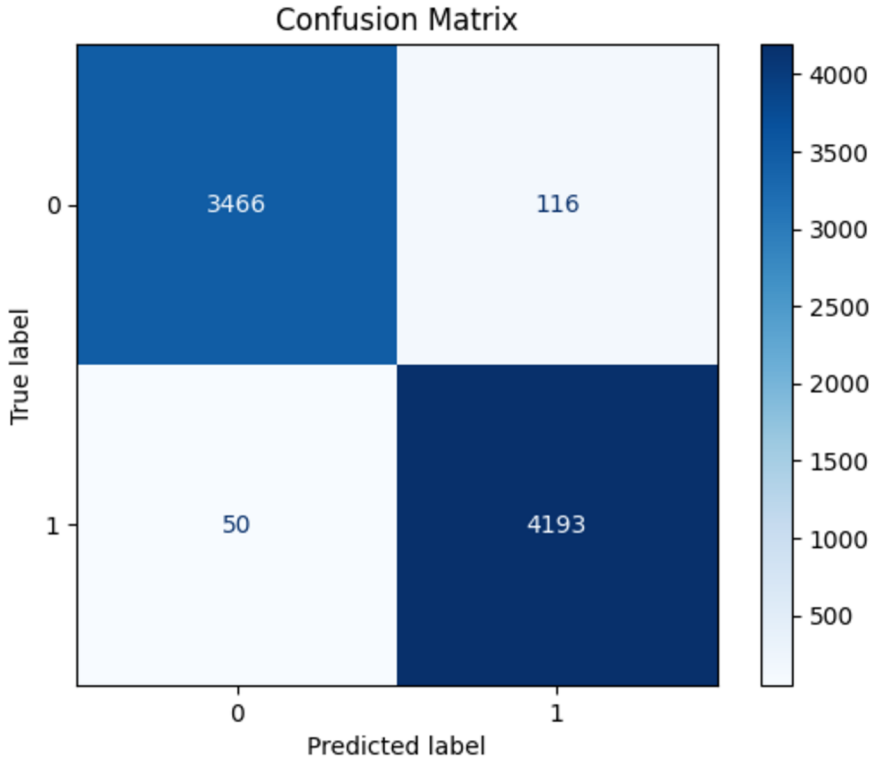
- Epoch 1: Train Loss = 0.2193, Val Loss = 0.0875
- Epoch 2: Train Loss = 0.0573, Val Loss = 0.0794
- Epoch 3: Train Loss = 0.0221, Val Loss = 0.1157
- Epoch 4: Train Loss = 0.0057, Val Loss = 0.1010
- Epoch 5: Train Loss = 0.0016, Val Loss = 0.0996
- Epoch 6: Train Loss = 0.0010, Val Loss = 0.1001
- Epoch 7: Train Loss = 0.0006, Val Loss = 0.1023
- Epoch 8: Train Loss = 0.0378, Val Loss = 0.0986
- Epoch 9: Train Loss = 0.0303, Val Loss = 0.0990
- Epoch 10: Train Loss = 0.0199, Val Loss = 0.0853



accuracy: 0.9788 - loss: 0.0815

here becauese all of values a above of 95% in accuracy and less than 0.20 in loss so the shape looks this way.

- Accuracy: 0.98
- Precision: 0.99 (class 0), 0.97 (class 1)
- Recall: 0.97 (class 0), 0.99 (class 1)
- F1-score: 0.98 (for both classes)



Class	Precision	Recall	F1-Score	Support
0 (Fake)	0.99	0.97	0.98	3582
1 (Real)	0.97	0.99	0.98	4243
Accuracy			**0.98**	7825
Macro Avg	0.98	0.98	0.98	7825
Weighted Avg	0.98	0.98	0.98	7825

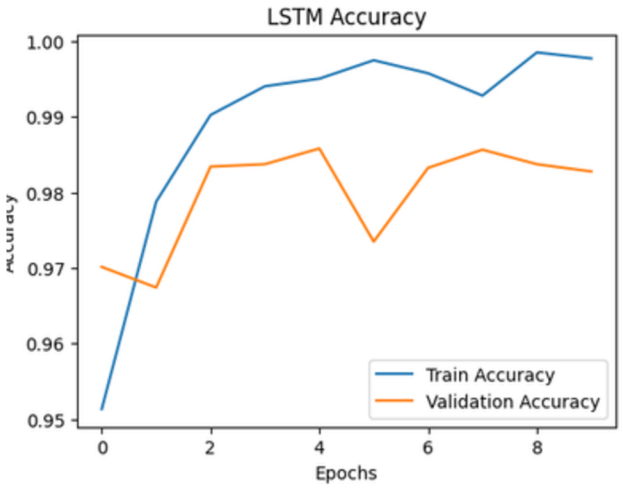
The Second model in sequential models is LSTM:

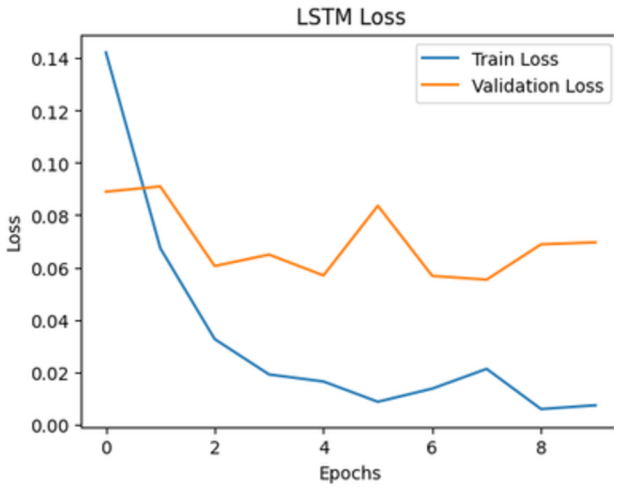
Training Results (10 Epochs):

- Epoch 1: Training Acc = 0.95, Val Acc = 0.97
- Epoch 2: Training Acc = 0.98, Val Acc = 0.97
- Epoch 3: Training Acc = 0.99, Val Acc = 0.98
- Epoch 4: Training Acc = 0.99, Val Acc = 0.98
- Epoch 5: Training Acc = 1.00, Val Acc = 0.99
- Epoch 6: Training Acc = 1.00, Val Acc = 0.97
- Epoch 7: Training Acc = 1.00, Val Acc = 0.98
- Epoch 8: Training Acc = 0.99, Val Acc = 0.99
- Epoch 9: Training Acc = 1.00, Val Acc = 0.98
- Epoch 10: Training Acc = 1.00, Val Acc = 0.98

Loss Results (10 Epochs)

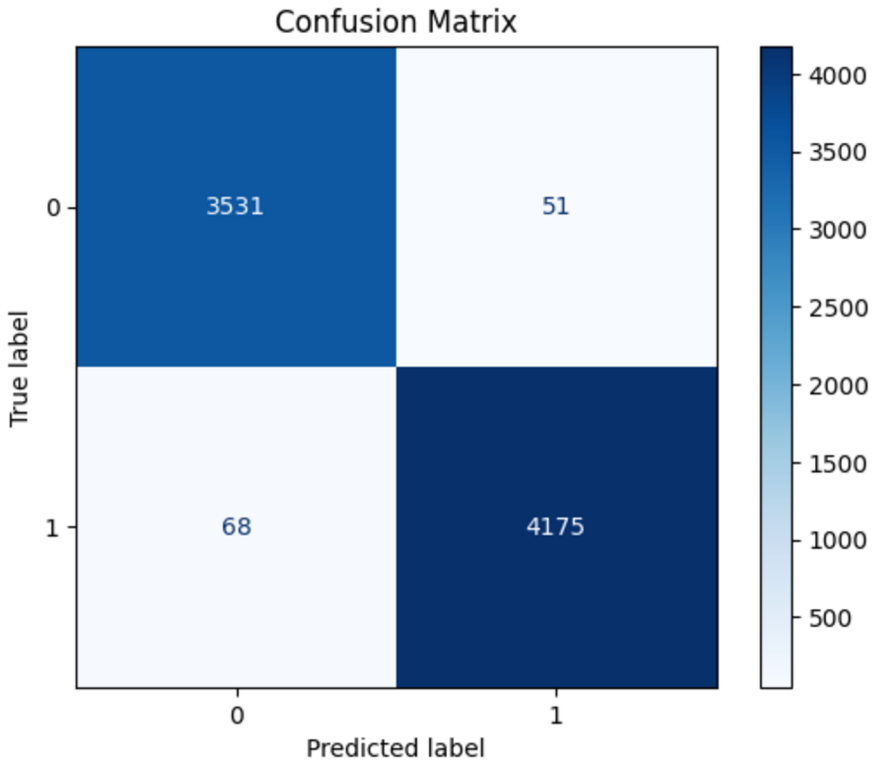
- Epoch 1: Train Loss = 0.1423, Val Loss = 0.0890
- Epoch 2: Train Loss = 0.0673, Val Loss = 0.0910
- Epoch 3: Train Loss = 0.0326, Val Loss = 0.0605
- Epoch 4: Train Loss = 0.0190, Val Loss = 0.0649
- Epoch 5: Train Loss = 0.0164, Val Loss = 0.0569
- Epoch 6: Train Loss = 0.0086, Val Loss = 0.0836
- Epoch 7: Train Loss = 0.0136, Val Loss = 0.0568
- Epoch 8: Train Loss = 0.0212, Val Loss = 0.0553
- Epoch 9: Train Loss = 0.0058, Val Loss = 0.0688
- Epoch 10: Train Loss = 0.0073, Val Loss = 0.0696





accuracy: 0.9788 - loss: 0.0815

- Accuracy: 0.98
- Precision: 0.98 (class 0), 0.99 (class 1)
- Recall: 0.99 (class 0), 0.98 (class 1)
- F1-score: 0.98 – 0.99



Class	Precision	Recall	F1-Score	Support
0 (Fake)	0.98	0.99	0.98	3582
1 (Real)	0.99	0.98	0.99	4243
Accuracy	–	–	0.98	7825
Macro Avg	0.98	0.98	0.98	7825
Weighted Avg	0.98	0.98	0.98	7825

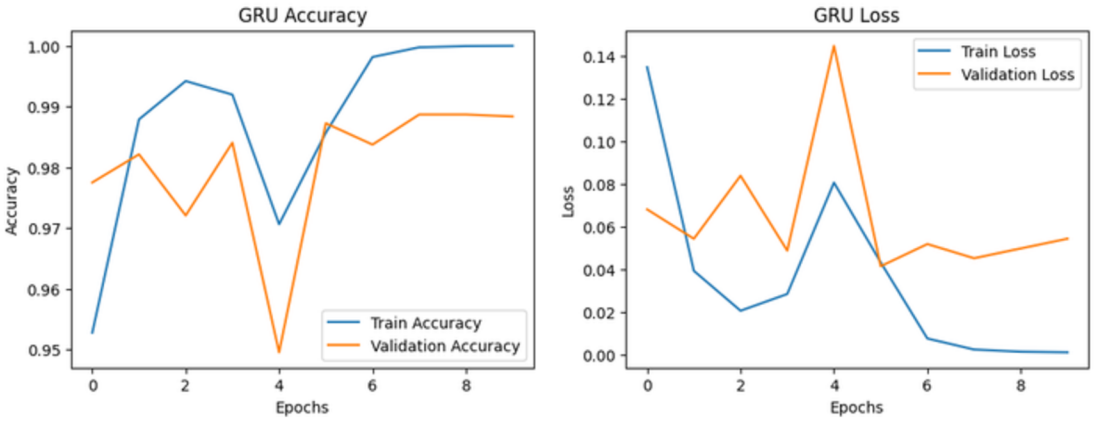
The Third model in sequential models is GRU:

Training Results (10 Epochs):

- Epoch 1: Training Acc = 0.95, Val Acc = 0.98
- Epoch 2: Training Acc = 0.99, Val Acc = 0.98
- Epoch 3: Training Acc = 0.99, Val Acc = 0.97
- Epoch 4: Training Acc = 0.99, Val Acc = 0.98
- Epoch 5: Training Acc = 0.97, Val Acc = 0.95
- Epoch 6: Training Acc = 0.99, Val Acc = 0.99
- Epoch 7: Training Acc = 1.00, Val Acc = 0.98
- Epoch 8: Training Acc = 1.00, Val Acc = 0.99
- Epoch 9: Training Acc = 1.00, Val Acc = 0.99
- Epoch 10: Training Acc = 1.00, Val Acc = 0.99

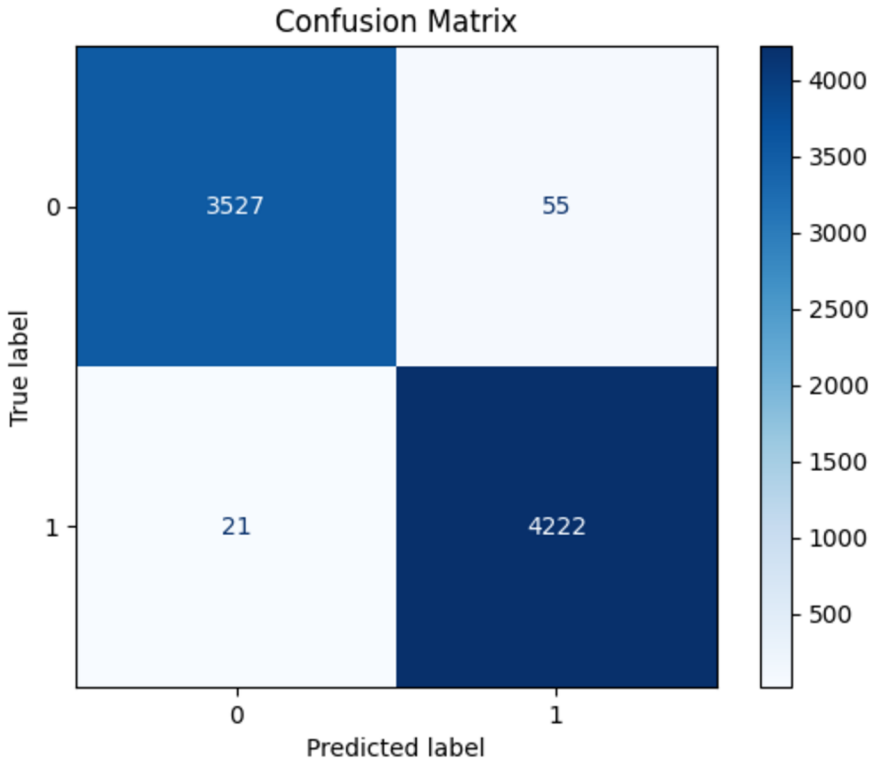
Loss Results (10 Epochs)

- Epoch 1: Train Loss = 0.1347, Val Loss = 0.0680
- Epoch 2: Train Loss = 0.0392, Val Loss = 0.0542
- Epoch 3: Train Loss = 0.0204, Val Loss = 0.0838
- Epoch 4: Train Loss = 0.0283, Val Loss = 0.0486
- Epoch 5: Train Loss = 0.0806, Val Loss = 0.1447
- Epoch 6: Train Loss = 0.0431, Val Loss = 0.0413
- Epoch 7: Train Loss = 0.0074, Val Loss = 0.0517
- Epoch 8: Train Loss = 0.0022, Val Loss = 0.0451
- Epoch 9: Train Loss = 0.0012, Val Loss = 0.0497
- Epoch 10: Train Loss = 0.0009, Val Loss = 0.0542



accuracy: 0.9903 - loss: 0.0454

- Accuracy: 0.99
- Precision: 0.99 (Class 0 – Fake), 0.99 (Class 1 – Real)
- Recall: 0.98 (Class 0 – Fake), 1.00 (Class 1 – Real)
- F1-Score: 0.99 – 0.99



Class	Precision	Recall	F1-Score	Support
0 (Fake)	0.99	0.98	0.99	3582
1 (Real)	0.99	1	0.99	4243
Accuracy			**0.99**	7825
Macro Avg	0.99	0.99	0.99	7825
Weighted Avg	0.99	0.99	0.99	7825

Then, I use a glove model for embedding data to a more accurate models.

Glove file: glove.6B.300d.txt

So I train three models again, and make after making preprocessing with Glove.

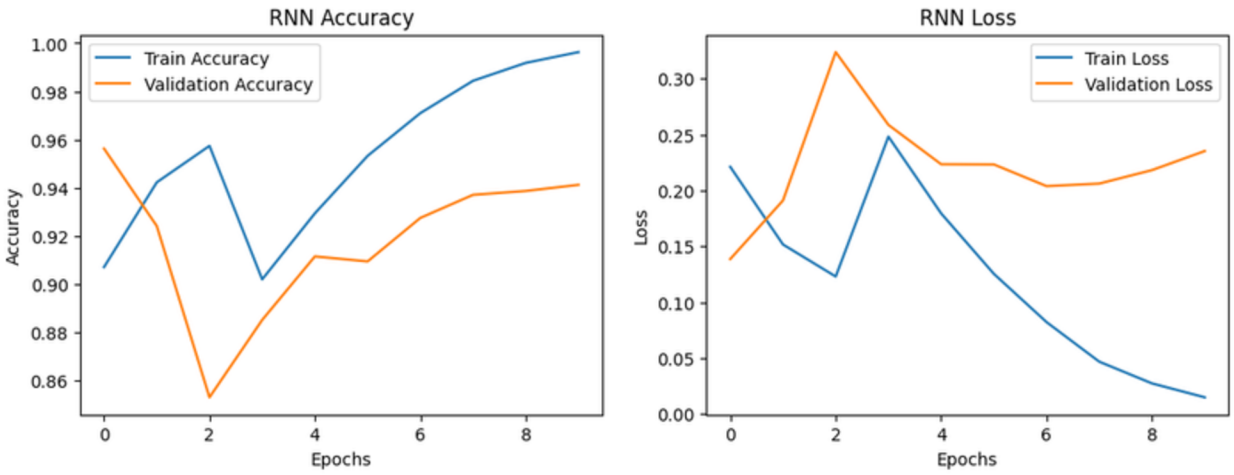
RNN With Glove:

Training Results (10 Epochs):

- Epoch 1: Training Acc = 0.91, Val Acc = 0.96
- Epoch 2: Training Acc = 0.94, Val Acc = 0.92
- Epoch 3: Training Acc = 0.96, Val Acc = 0.85
- Epoch 4: Training Acc = 0.90, Val Acc = 0.89
- Epoch 5: Training Acc = 0.93, Val Acc = 0.91
- Epoch 6: Training Acc = 0.95, Val Acc = 0.91
- Epoch 7: Training Acc = 0.97, Val Acc = 0.93
- Epoch 8: Training Acc = 0.98, Val Acc = 0.94
- Epoch 9: Training Acc = 0.99, Val Acc = 0.94
- Epoch 10: Training Acc = 0.99, Val Acc = 0.94

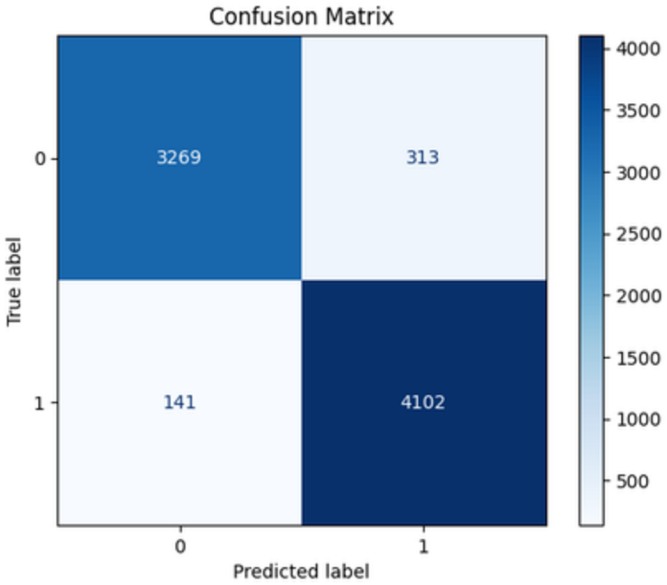
Loss Results (10 Epochs)

- Epoch 1: Train Loss = 0.22, Val Loss = 0.14
- Epoch 2: Train Loss = 0.15, Val Loss = 0.19
- Epoch 3: Train Loss = 0.12, Val Loss = 0.32
- Epoch 4: Train Loss = 0.25, Val Loss = 0.26
- Epoch 5: Train Loss = 0.18, Val Loss = 0.22
- Epoch 6: Train Loss = 0.13, Val Loss = 0.22
- Epoch 7: Train Loss = 0.08, Val Loss = 0.20
- Epoch 8: Train Loss = 0.05, Val Loss = 0.21
- Epoch 9: Train Loss = 0.03, Val Loss = 0.22
- Epoch 10: Train Loss = 0.02, Val Loss = 0.24



accuracy: 0.9397 - loss: 0.2438

- Accuracy: 0.94
- Precision: 0.96 (class 0), 0.93 (class 1)
- Recall: 0.91 (class 0), 0.97 (class 1)
- F1-score: 0.94 – 0.95



Class	Precision	Recall	F1-Score	Support
0 (Fake)	0.96	0.91	0.94	3582
1 (Real)	0.93	0.97	0.95	4243
Accuracy			**0.94**	7825
Macro Avg	0.94	0.94	0.94	7825
Weighted Avg	0.94	0.94	0.94	7825

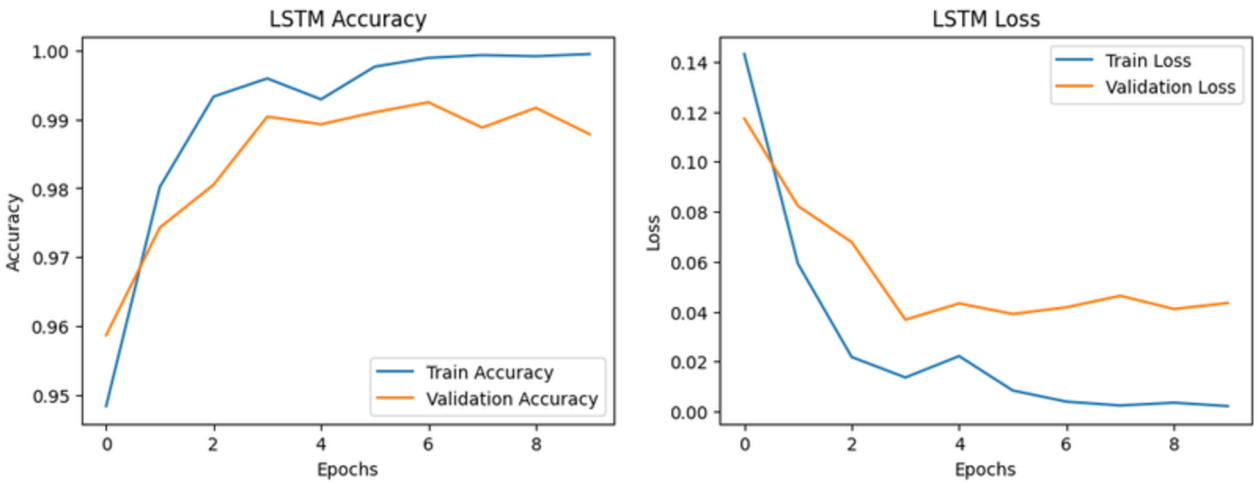
LSTM With Glove:

Training Results (10 Epochs):

- Epoch 1: Training Acc = 0.95, Val Acc = 0.96
- Epoch 2: Training Acc = 0.98, Val Acc = 0.97
- Epoch 3: Training Acc = 0.99, Val Acc = 0.98
- Epoch 4: Training Acc = 1.00, Val Acc = 0.99
- Epoch 5: Training Acc = 0.99, Val Acc = 0.99
- Epoch 6: Training Acc = 1.00, Val Acc = 0.99
- Epoch 7: Training Acc = 1.00, Val Acc = 0.99
- Epoch 8: Training Acc = 1.00, Val Acc = 0.99
- Epoch 9: Training Acc = 1.00, Val Acc = 0.99
- Epoch 10: Training Acc = 1.00, Val Acc = 0.99

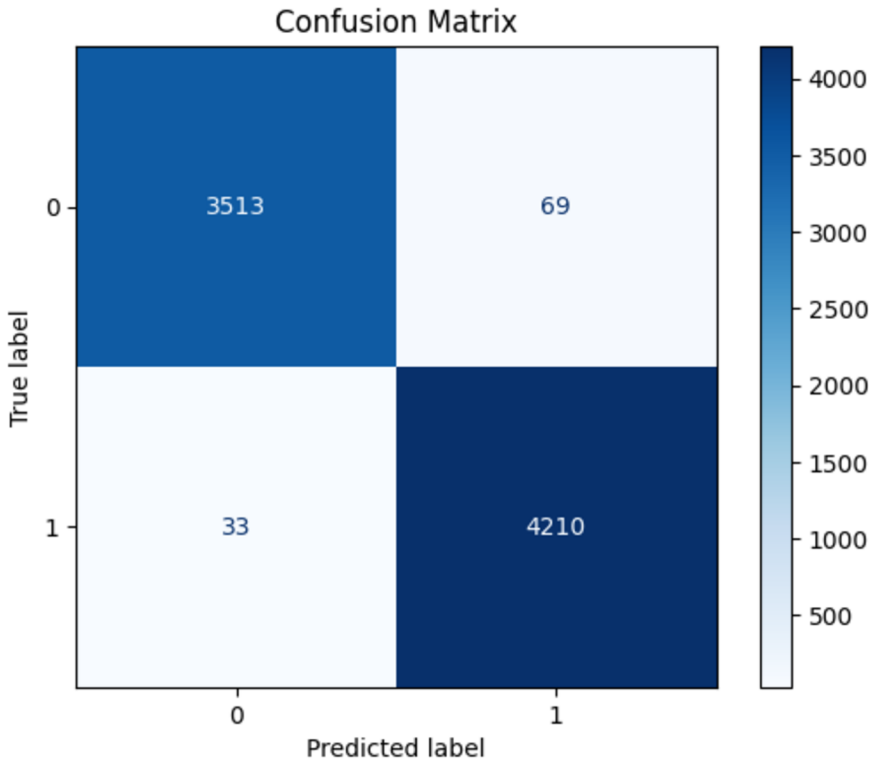
Loss Results (10 Epochs)

- Epoch 1: Train Loss = 0.1431, Val Loss = 0.1173
- Epoch 2: Train Loss = 0.0591, Val Loss = 0.0823
- Epoch 3: Train Loss = 0.0218, Val Loss = 0.0678
- Epoch 4: Train Loss = 0.0136, Val Loss = 0.0368
- Epoch 5: Train Loss = 0.0222, Val Loss = 0.0433
- Epoch 6: Train Loss = 0.0084, Val Loss = 0.0390
- Epoch 7: Train Loss = 0.0040, Val Loss = 0.0417
- Epoch 8: Train Loss = 0.0024, Val Loss = 0.0463
- Epoch 9: Train Loss = 0.0035, Val Loss = 0.0410
- Epoch 10: Train Loss = 0.0022, Val Loss = 0.0434



accuracy: 0.9885 - loss: 0.0369

- Accuracy: 0.99
- Precision: 0.99 (class 0), 0.98 (class 1)
- Recall: 0.98 (class 0), 0.99 (class 1)
- F1-score: 0.99



Class	Precision	Recall	F1-Score	Support
0 (Fake)	0.99	0.98	0.99	3582
1 (Real)	0.98	0.99	0.99	4243
Accuracy			**0.99**	7825
Macro Avg	0.99	0.99	0.99	7825
Weighted Avg	0.99	0.99	0.99	7825

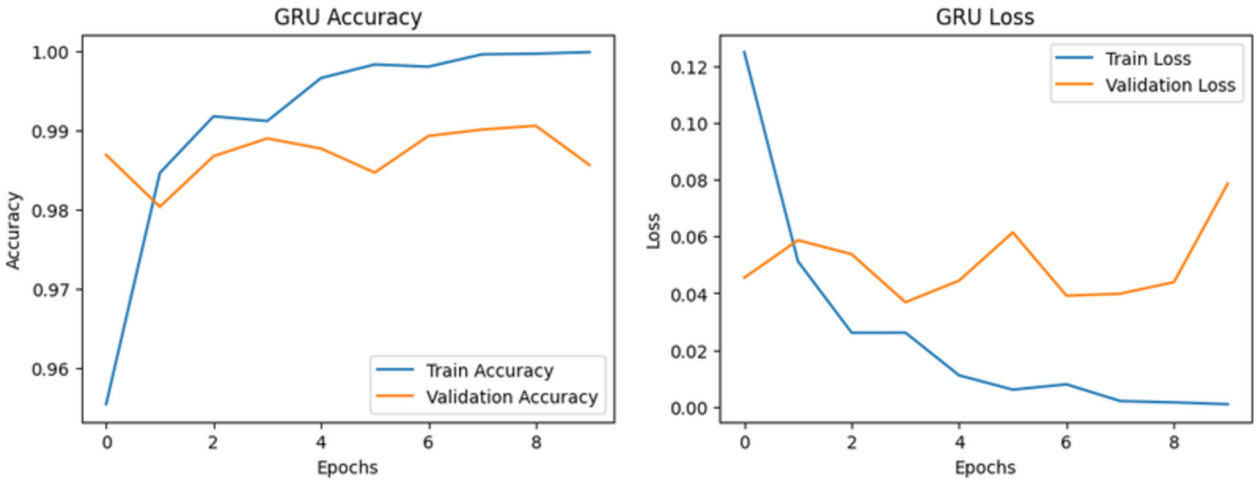
GRU With Glove:

Training Results (10 Epochs):

- Epoch 1: Training Acc = 0.96, Val Acc = 0.99
- Epoch 2: Training Acc = 0.98, Val Acc = 0.98
- Epoch 3: Training Acc = 0.99, Val Acc = 0.99
- Epoch 4: Training Acc = 0.99, Val Acc = 0.99
- Epoch 5: Training Acc = 1.00, Val Acc = 0.99
- Epoch 6: Training Acc = 1.00, Val Acc = 0.98
- Epoch 7: Training Acc = 1.00, Val Acc = 0.99
- Epoch 8: Training Acc = 1.00, Val Acc = 0.99
- Epoch 9: Training Acc = 1.00, Val Acc = 0.99
- Epoch 10: Training Acc = 1.00, Val Acc = 0.99

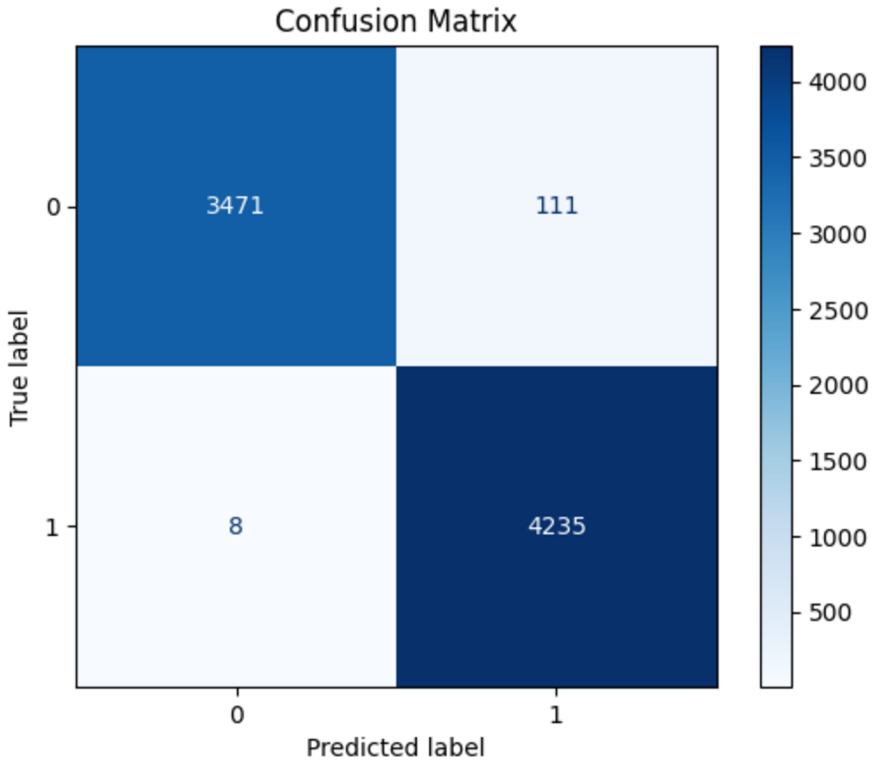
Loss Results (10 Epochs)

- Epoch 1: Train Loss = 0.1249, Val Loss = 0.0454
- Epoch 2: Train Loss = 0.0512, Val Loss = 0.0587
- Epoch 3: Train Loss = 0.0261, Val Loss = 0.0537
- Epoch 4: Train Loss = 0.0261, Val Loss = 0.0368
- Epoch 5: Train Loss = 0.0111, Val Loss = 0.0444
- Epoch 6: Train Loss = 0.0060, Val Loss = 0.0614
- Epoch 7: Train Loss = 0.0079, Val Loss = 0.0391
- Epoch 8: Train Loss = 0.0020, Val Loss = 0.0398
- Epoch 9: Train Loss = 0.0016, Val Loss = 0.0439
- Epoch 10: Train Loss = 0.0009, Val Loss = 0.0786



accuracy: 0.9885 - loss: 0.0369

- Accuracy: 0.99
- Precision: 0.99 (class 0), 0.98 (class 1)
- Recall: 0.98 (class 0), 0.99 (class 1)
- F1-score: 0.99



Class	Precision	Recall	F1-Score	Support
0 (Fake)	1	0.97	0.98	3582
1 (Real)	0.97	1	0.99	4243
Accuracy			**0.98**	7825
Macro Avg	0.99	0.98	0.98	7825
Weighted Avg	0.99	0.98	0.98	7825

Among the three models, Glove preprocessed the LSTM model achieved the highest performance with an accuracy of 99%, with strong precision, recall, and F1-score. The GRU model performed very closely, with 98% accuracy, and is slightly more lightweight and faster to train than LSTM. The basic RNN model achieved only 94% accuracy, making it significantly weaker compared to LSTM and GRU.

LSTM is the best choice for maximum accuracy, while GRU is a strong alternative if faster training and simpler architecture are preferred.

4.Tranformer and Fine-Tuning:

I made preprocessing of the data, then I started to build a transformer-based DistilBERT.
Then perform Fine-tuning model and save it to use in the web app.

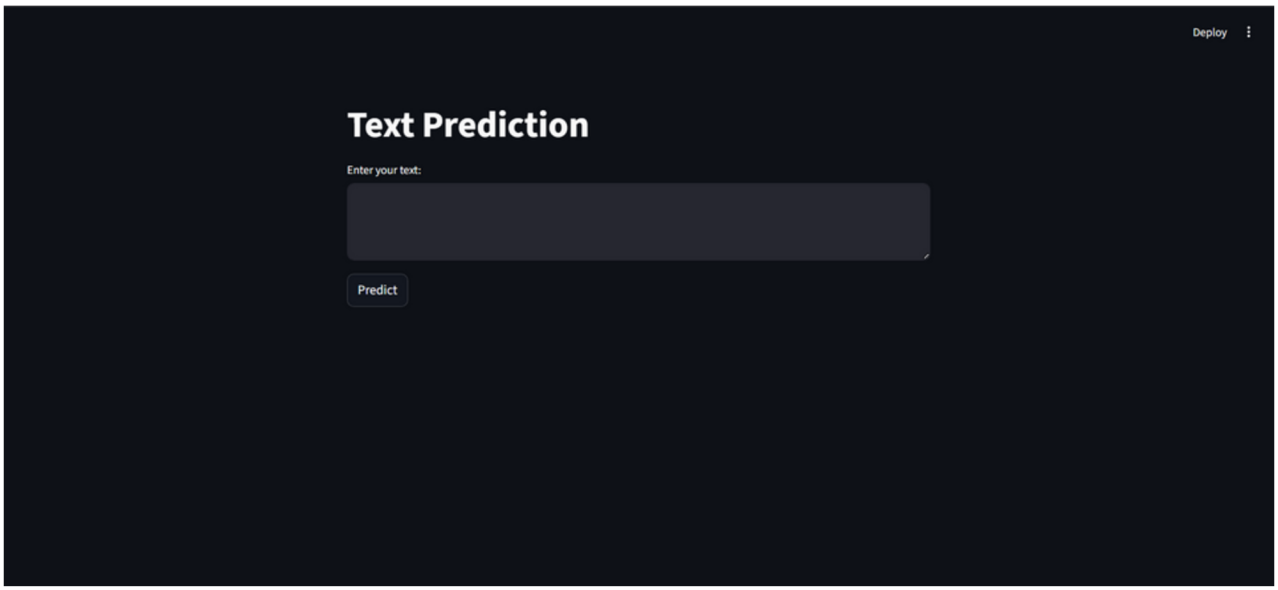
Epoch	Training Loss	Validation Loss
1	0.0141	0.000202
2	0.0006	0.00004
3	0.0003	0.000029

Fine-tuning metrics:

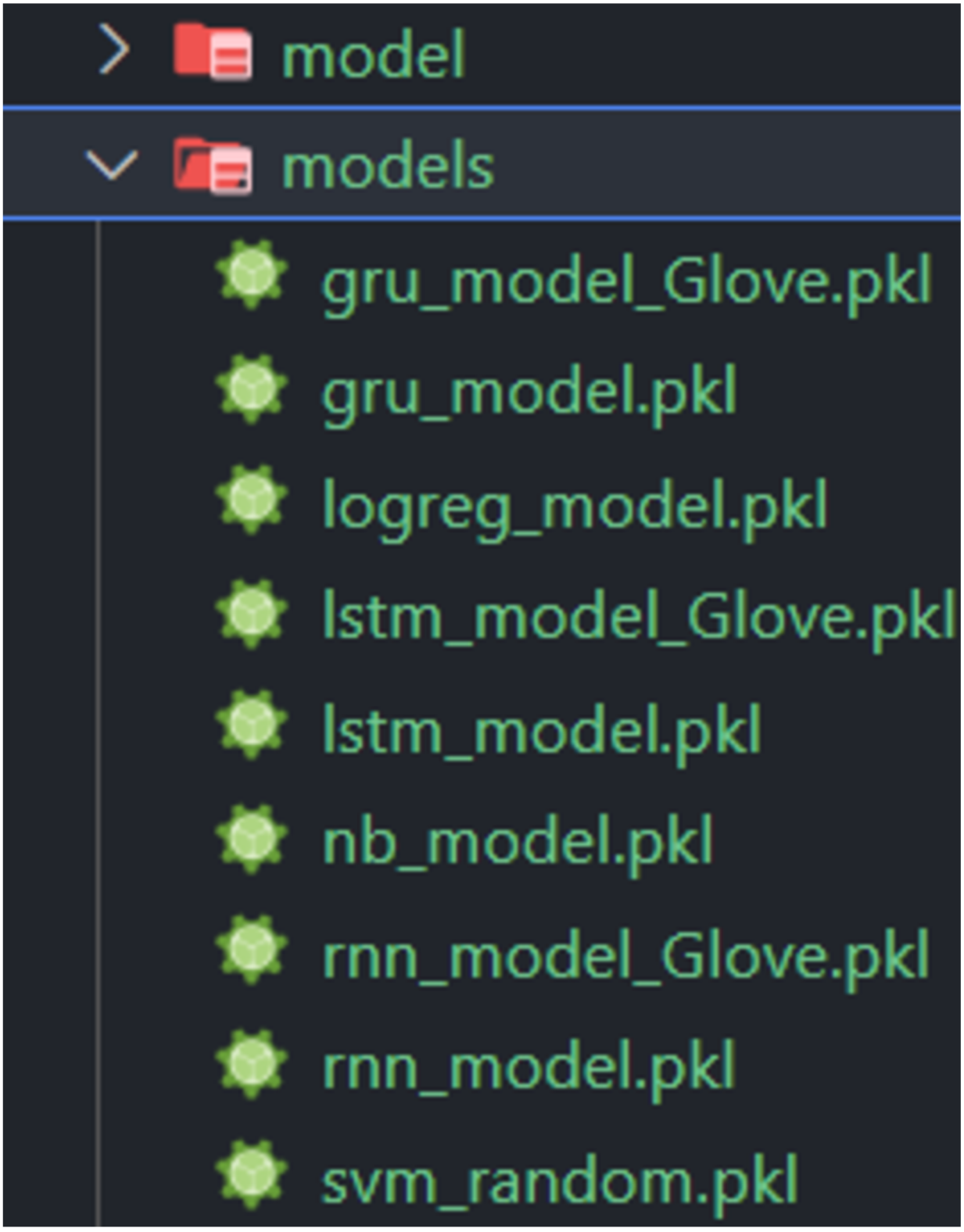
Metric	Value
Evaluation Loss	0.0000291
Accuracy	1
Precision	1
Recall	1
F1-score	1
Runtime (s)	43.54
Samples per Second	205.29
Steps per Second	6.43

The Transformer model, after fine-tuning, achieved outstanding performance with perfect scores across all evaluation metrics (Accuracy, Precision, Recall, and F1-score = 1.0). The extremely low evaluation loss (2.91e-05) indicates excellent generalization and stability of the model. Compared to traditional RNN-based architectures such as LSTM and GRU, the fine-tuned Transformer clearly demonstrates superior performance, benefiting from its attention mechanism and ability to capture long-range dependencies effectively. This confirms that fine-tuning Transformer-based models is a highly effective approach for achieving state-of-the-art results in text classification tasks.

Finally, I made a steamlit app to predict the text with the model eather True class or Fake class.



And saved any model I had trained in a pkl file to reuse in the future.



Error Analysis

Although the Transformer model achieved almost perfect results (Accuracy, Precision, Recall, and F1 = 1.0), it is important to consider possible sources of error:

Data distribution bias: If the dataset is not fully balanced or diverse, the model might overfit and fail when exposed to unseen or real-world data.

Overfitting risks: Extremely low validation loss and perfect scores may suggest overfitting to the training/validation set. Testing on an independent dataset is necessary to validate robustness.

Edge cases: The model may struggle with ambiguous, noisy, or adversarial inputs that were underrepresented in the training data.

Ethical Concerns

When deploying Transformer-based models, several ethical issues must be considered:

Bias and Fairness: If the training data reflects social or demographic biases, the model may reproduce or even amplify these biases in predictions.

Privacy: Using sensitive or personally identifiable information in training could raise privacy risks if not handled properly.

Misinformation: Highly accurate models can be misused to generate or spread convincing fake content (e.g., fake news or harmful text).

Transparency & Accountability: Lack of explainability may reduce trust and accountability in critical applications.