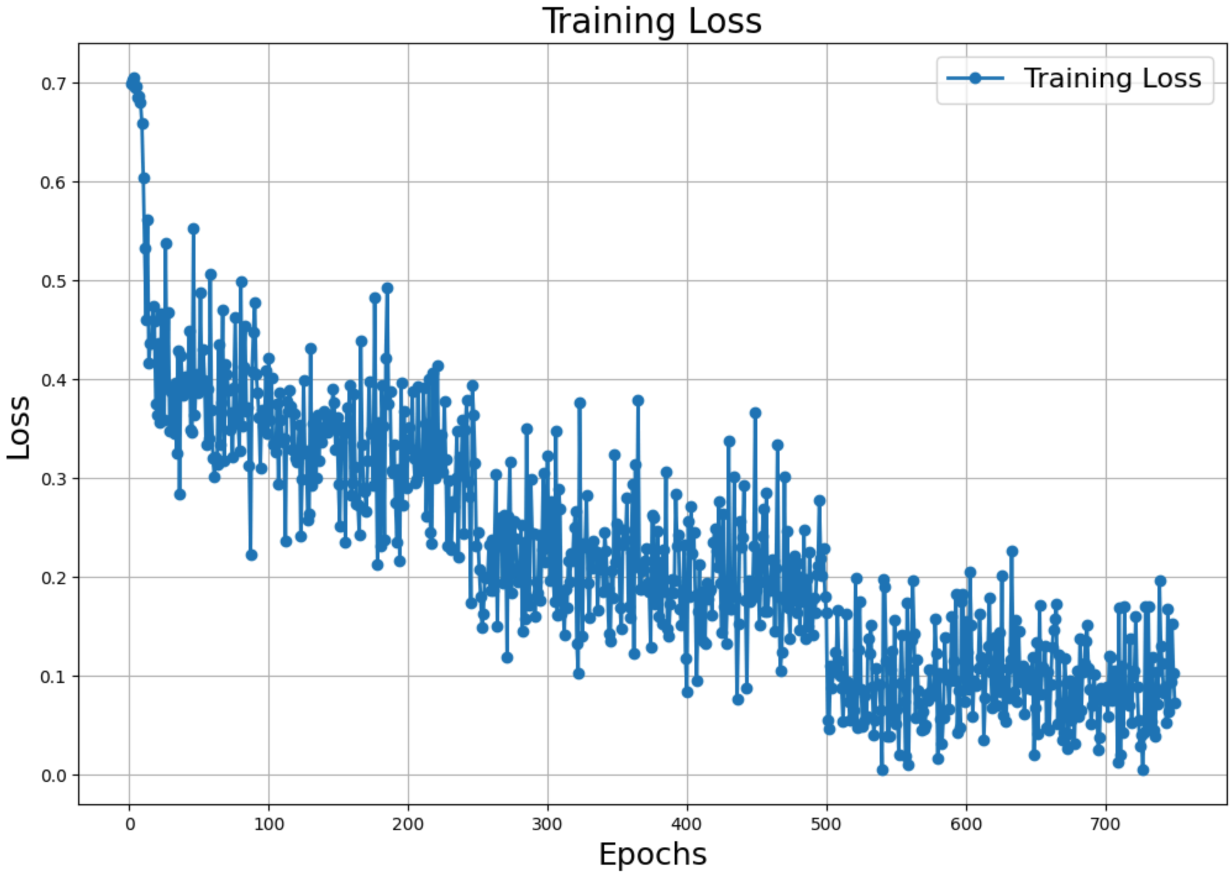
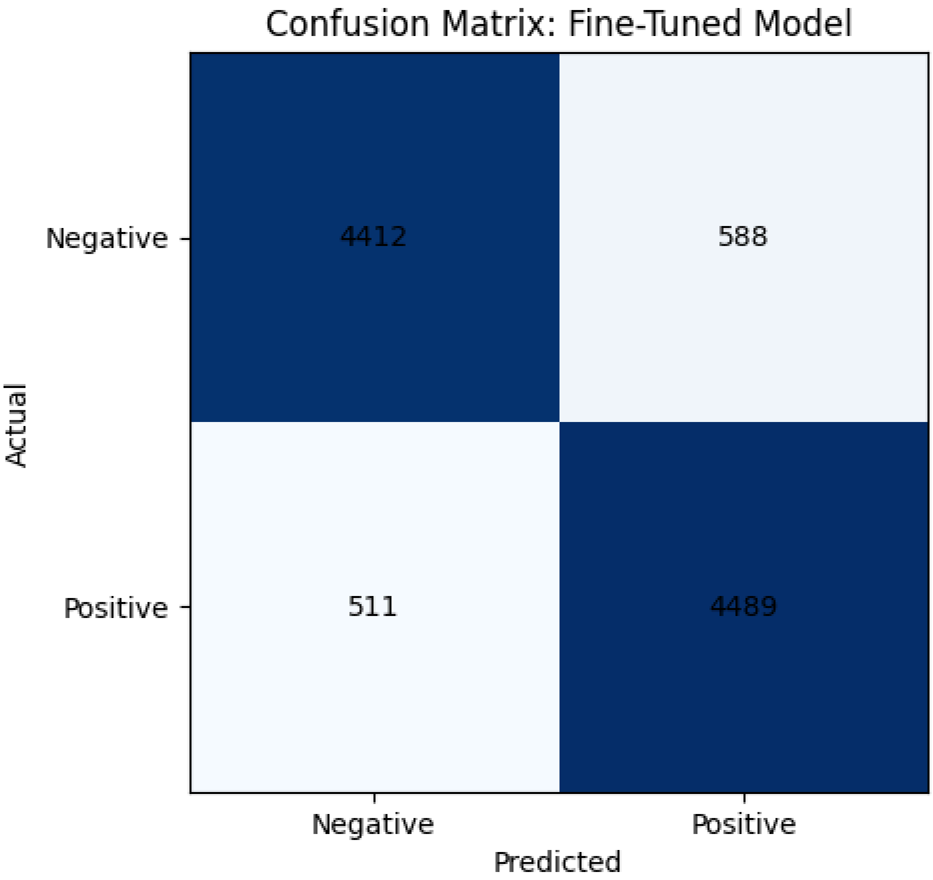
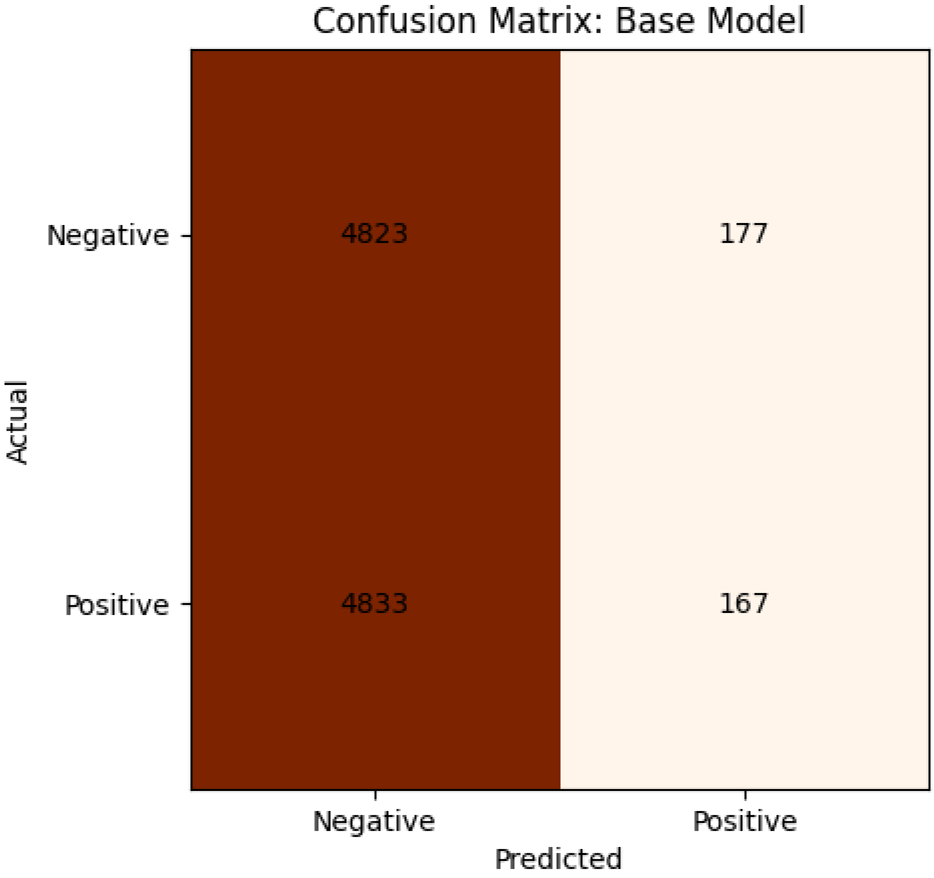
Information about my code:
The imdb file is assumed to be in a data folder called “data”.
Here is my github repo link:
https://github.com/waleed3114/CSCE580-Fall2024-WaleedKhan-Repo/blob/main/imdbclassificationprojectWaleedKhan.ipynb

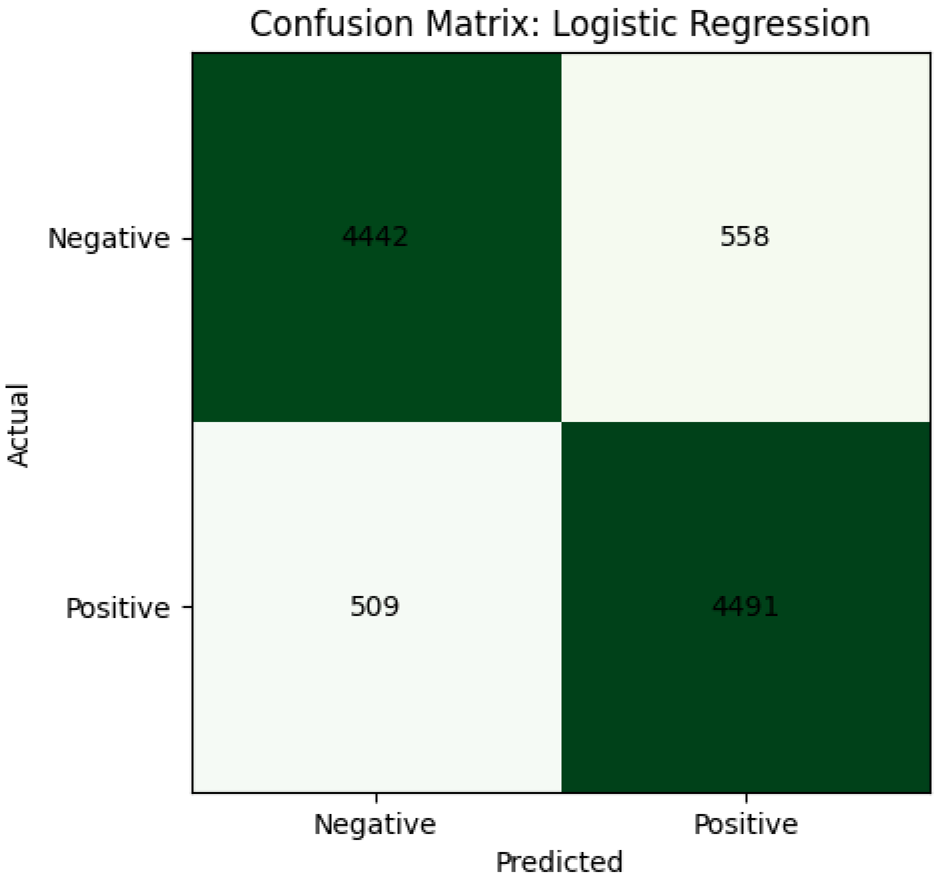
1. Accuracy and Loss Curves



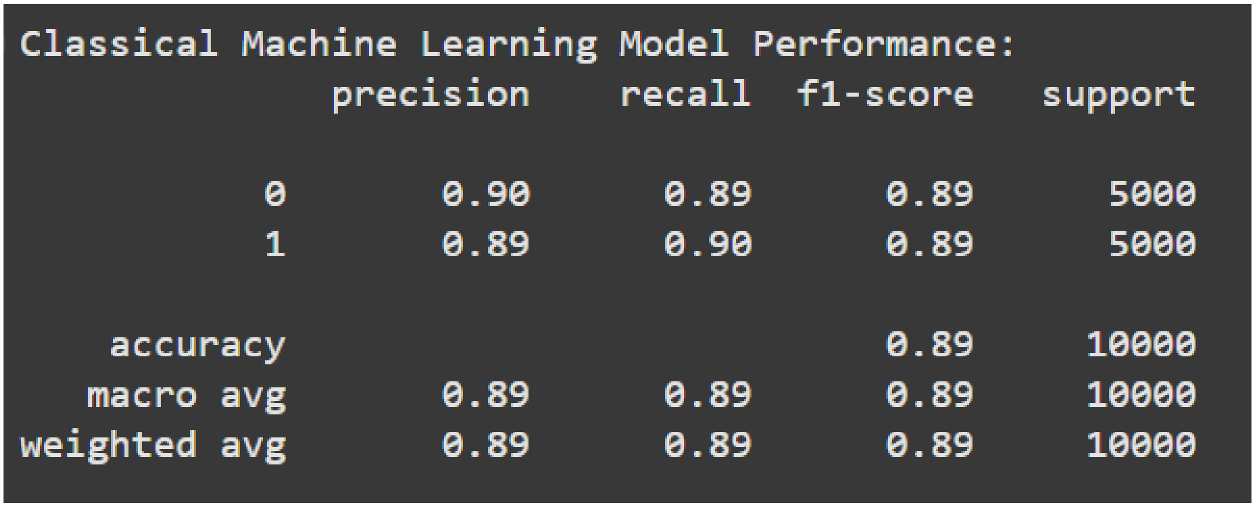
2. Confusion Matrix


Confusion Matrix: Fine-Tuned Model
Negative
Positive
Actual
4412
588
511
4489
Negative
Positive
Predicted

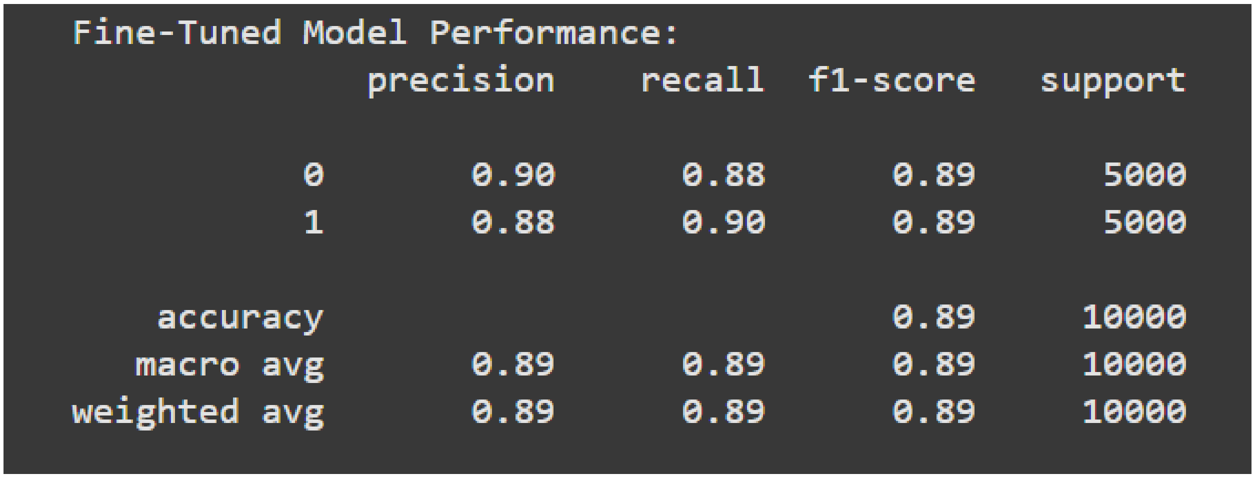

Confusion Matrix: Base Model
Actual
Negative
Positive
4823
177
4833
167
Negative
Positive
Predicted



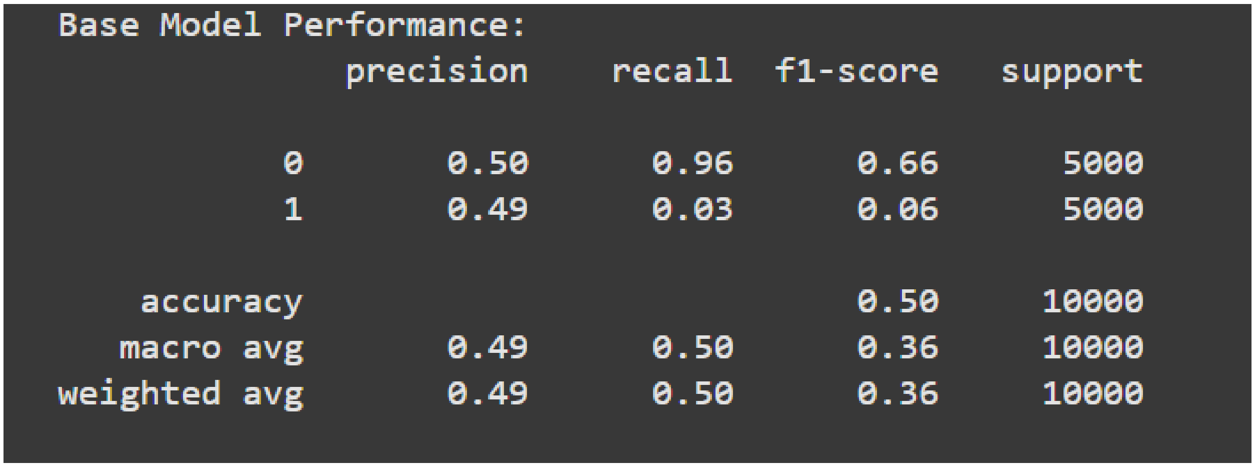
3. Precision, Recall, and F1- Score



```
Classical Machine Learning Model Performance:
              precision    recall  f1-score   support

           0       0.90      0.89      0.89      5000
           1       0.89      0.90      0.89      5000

    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000
```

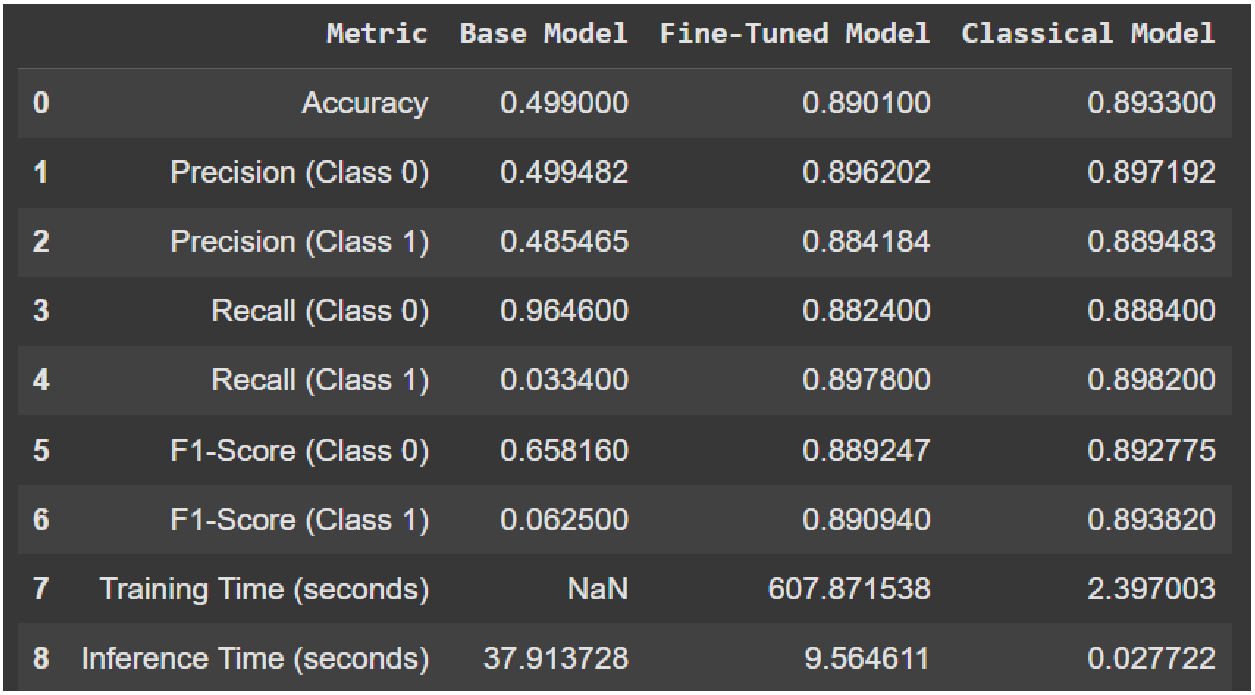


```
Fine-Tuned Model Performance:
              precision    recall  f1-score   support

           0       0.90      0.88      0.89      5000
           1       0.88      0.90      0.89      5000

    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000
```



```
Base Model Performance:
              precision    recall  f1-score   support

           0       0.50      0.96      0.66      5000
           1       0.49      0.03      0.06      5000

    accuracy                           0.50     10000
   macro avg       0.49      0.50      0.36     10000
weighted avg       0.49      0.50      0.36     10000
```

4. Performance Comparison and time complexity



	Metric	Base Model	Fine-Tuned Model	Classical Model
0	Accuracy	0.499000	0.890100	0.893300
1	Precision (Class 0)	0.499482	0.896202	0.897192
2	Precision (Class 1)	0.485465	0.884184	0.889483
3	Recall (Class 0)	0.964600	0.882400	0.888400
4	Recall (Class 1)	0.033400	0.897800	0.898200
5	F1-Score (Class 0)	0.658160	0.889247	0.892775
6	F1-Score (Class 1)	0.062500	0.890940	0.893820
7	Training Time (seconds)	NaN	607.871538	2.397003
8	Inference Time (seconds)	37.913728	9.564611	0.027722

Questions:
1. What do the accuracy and loss curves tell you about the fine-tuning process?

The accuracy and loss curves tell us that the data loss decreases over time with respect to the epochs. This means that the model is learning effectively and is improving.

2. How does the fine-tuned DistilBERT model compare to the classical ML model? What advantages or limitations do transformers present over classical algorithms?

The fine-tuned DistilBERT had similar accuracy and precision rates as the logistic regression classical model. This was because the logistic regression model was trained enough for them to have similar values for these parameters and this task wasn't difficult enough for the transformer to really be much better than the logistic regression. Transformers have the advantage of being good at working with complex data, but they can sometimes be overfitting for simple data sets and have the additional drawback of being heavily resource dependent.

3. What insights can you draw from the confusion matrix? Are there any patterns in the misclassifications?
I can draw the insight that logistic regression has the potential to match the performance of fine tuned models for simple datasets. Also, the models had some trouble capturing negative (0) values at times.

4. Why might the fine-tuned model outperform the base model?
It might outperform the base model because the fine-tuned model is trained with a specific dataset. In the case of the base model, the base model is just trained with a generalized data set. The fine-tuned one is problem-specific and hence better-suited for our problem.

5. Which model would you recommend for deployment in a real-world scenario, and why? Consider both performance and efficiency in your answer.

For a complex dataset, I would recommend using the pretrained model and for less complex datasets I would prefer a classical machine learning model, like logistic regression or random forest. I feel like the pretrained models are typically better at solving real-world problems, which tend to be complex. However, it needs a lot of resources and time so it's not the best choice for a simple problem in terms of efficiency. The classical machine learning algorithm is better when it comes to finding something that's one size fits all and is good for smaller and simpler datasets.