SENTIMENT ANALYSIS: A COMPARATIVE STUDY OF TRADITIONAL & DEEP LEARNING APPROACHES

CONTRIBUTORS:

ABUBAKAR AHMED

BERNICE AKUDBILLA

PETER JOHNSON

PURITY KIHIU

GITHUB LINK: CLICK

1. Introduction

In a world saturated with opinions, from social media rants to customer reviews, understanding sentiment at scale has become a necessity. Sentiment analysis, also known as opinion mining, is the process of using computational techniques to detect emotional tone in textual content [6]. This technology plays a crucial role in business intelligence, brand monitoring, and social science research.

In this project, our goal was to build and evaluate a sentiment classification system using both a traditional machine learning model (Logistic Regression) and several deep learning models (LSTM variants). By comparing these models, we sought to determine which approach is better suited for binary sentiment classification in terms of both performance and practical applicability.

2. Literature Review

Early sentiment classification research predominantly relied on traditional models like Support Vector Machines (SVM), Naive Bayes, and Logistic Regression [7]. These models, while efficient and interpretable, often fall short when faced with complex sentence structures or subtle context.

Deep learning has since evolved, particularly through the use of Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Gated Recurrent Units (GRUs), which capture long-range dependencies in textual data.

Furthermore, feature representation has evolved from simple bag-of-words to rich vector embeddings. Traditional models benefit from TF-IDF representations, while deep learning leverages Word2Vec, GloVe, or trainable embeddings to enhance semantic understanding [8].

3. Methodology

3.1 Dataset Selection

We used the IMDB Movie Reviews 50K dataset available on Kaggle, which contains 50,000 pre-labeled movie reviews split into training and testing sets. Each review is labeled as either positive or negative.

3.2 Preprocessing

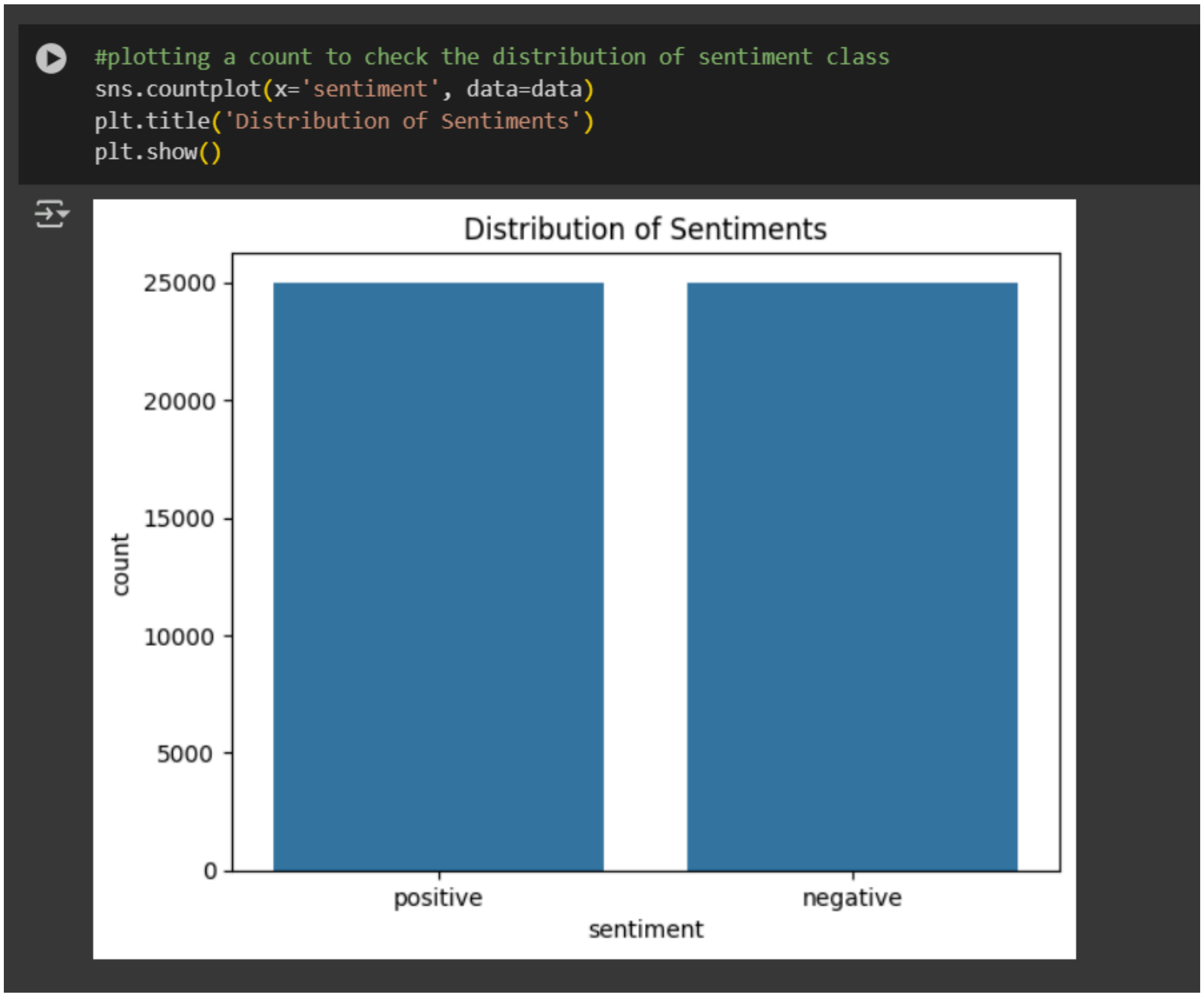
- Lowercased all text and removed punctuation
- Tokenized and removed stopwords
- Padded sequences to a uniform length of 200 tokens
- Applied TF-IDF vectorization for Logistic Regression
- Used a Keras tokenizer and embedding layer for deep learning models

3.3 Model Architectures

- **Logistic Regression**: Used TF-IDF features (200 dimensions)
- **Deep Learning (LSTM)**:
    - Embedding layer: vocab size = 5000, output_dim = 128
    - LSTM layers: one or two layers with dropout
    - Dense layers: one or two layers ending in a sigmoid for binary classification

3.4 Hyperparameter Tuning

- **Optimizers**: Adam, Nadam, RMSprop
- **Dropout rates**: 0.2, 0.3, 0.4
- **Epochs**: 10, 15, 20
- **Batch Size**: 64



```
#plotting a count to check the distribution of sentiment class
sns.countplot(x='sentiment', data=data)
plt.title('Distribution of Sentiments')
plt.show()
```

Fig 1.1 A plot showcasing the distribution of sentiment after pre-processing

4. Experiment Design

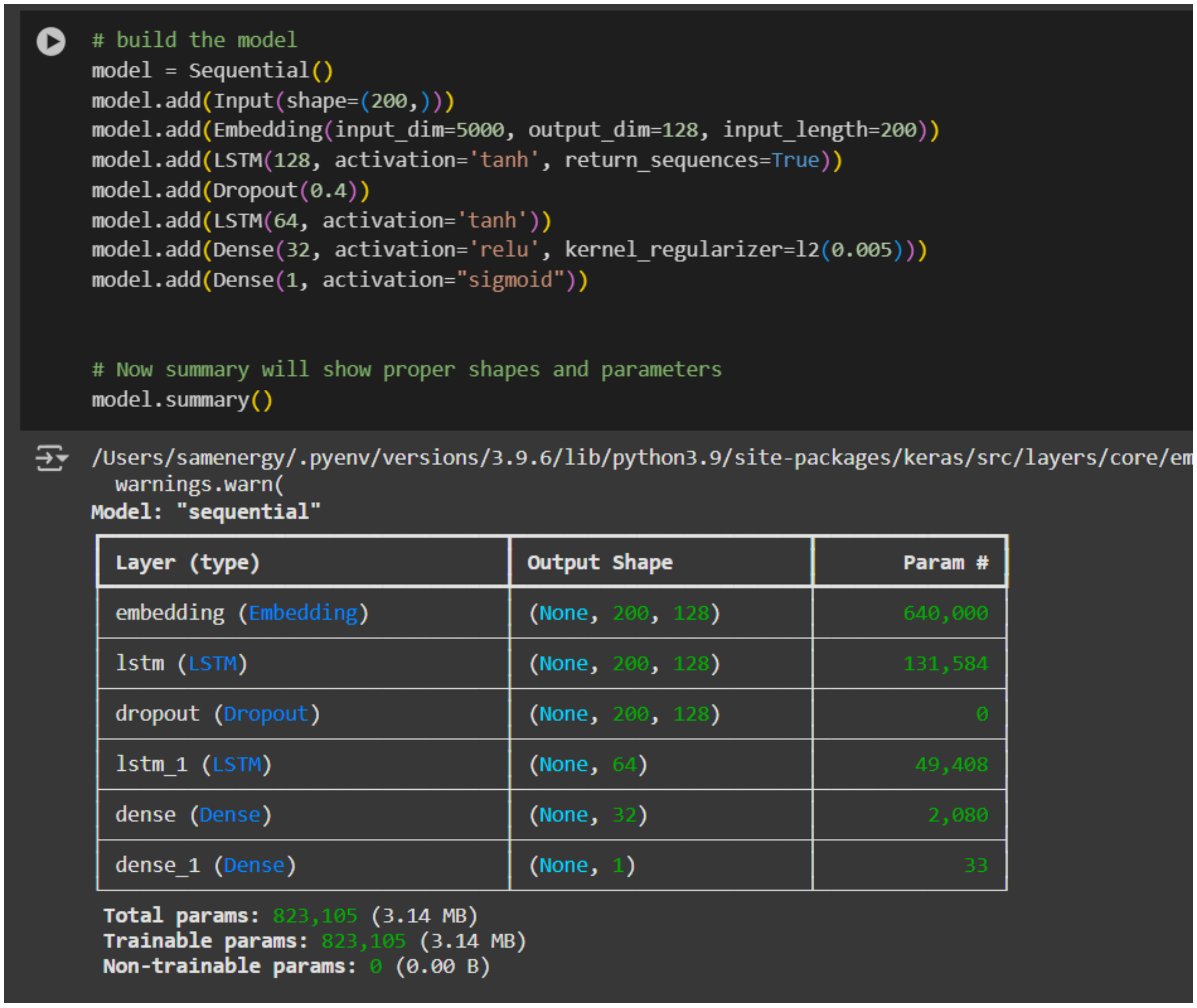
To understand the model's behavior, we did multiple experiments and checked their results in structured tables. The ones labeled with (D) have dropouts.

Table 1: Optimizer Comparison

Model ID	Optimizer	Drop out	Epochs	Accuracy	Notes
DL-001	Adam	0.3	10	0.8720	Initial baseline model with strong generalization
DL-002	Adam (D)	0.2	15	0.8618	Showed lower F1 compared to other models
DL-003	Nadam	0.4	20	0.8818	Best-performing model with high F1 and precision
DL-004	RMSprop	0.2	20	0.8688	Strong contender, slightly edged out by DL-005
DL-005	Nadam (D)	0.2	20	0.8679	Originally a strong candidate, not the top performer

Table 2: Traditional vs Deep Learning

Model	Accuracy	Precision	Recall	F1 Score
DL-003 (Nadam)	0.8818	0.8949	0.8672	0.8809
DL-001 (Adam)	0.8720	0.8742	0.8714	0.8728
DL-004 (RMSprop)	0.8688	0.8743	0.8639	0.8690
DL-005 (Nadam)	0.8679	0.8756	0.8601	0.8678
DL-002 (Adam)	0.8618	0.8544	0.8748	0.8645
Logistic Regression	0.86	0.85	0.87	0.86



```
# build the model
model = Sequential()
model.add(Input(shape=(200,)))
model.add(Embedding(input_dim=5000, output_dim=128, input_length=200))
model.add(LSTM(128, activation='tanh', return_sequences=True))
model.add(Dropout(0.4))
model.add(LSTM(64, activation='tanh'))
model.add(Dense(32, activation='relu', kernel_regularizer=l2(0.005)))
model.add(Dense(1, activation="sigmoid"))


# Now summary will show proper shapes and parameters
model.summary()
```

```
/Users/samenergy/.pyenv/versions/3.9.6/lib/python3.9/site-packages/keras/src/layers/core/em
  warnings.warn(
Model: "sequential"
```

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 200, 128)	640,000
lstm (LSTM)	(None, 200, 128)	131,584
dropout (Dropout)	(None, 200, 128)	0
lstm_1 (LSTM)	(None, 64)	49,408
dense (Dense)	(None, 32)	2,080
dense_1 (Dense)	(None, 1)	33

```
Total params: 823,105 (3.14 MB)
Trainable params: 823,105 (3.14 MB)
Non-trainable params: 0 (0.00 B)
```

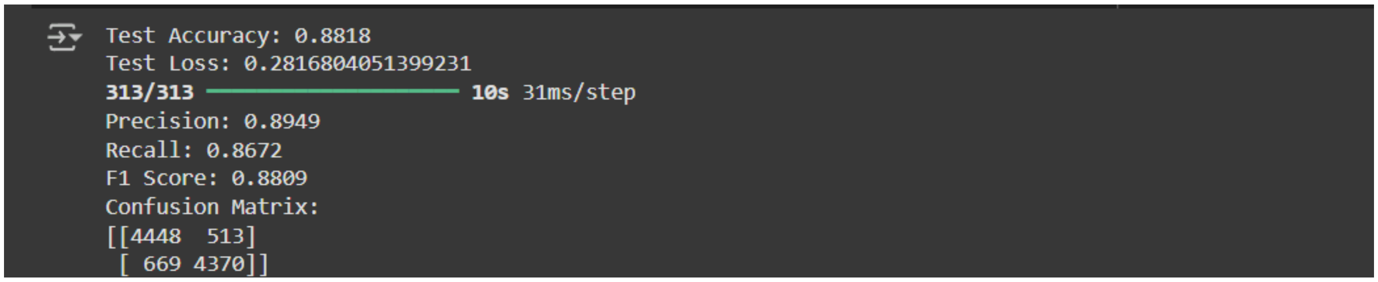
Fig 1.2 The Best Performing Model's Architecture (DL - 003)

5. Results & Discussion

The Logistic Regression model provided a solid baseline with an F1-score of 0.86, but it lacked the balance needed for more context in rich sentiment cases. While DL-005 (LSTM + Nadam + Dropout 0.2) showed commendable results with an F1-score of 0.8678, DL-003 (LSTM + Nadam + Dropout 0.4) ultimately delivered the best performance. It achieved the highest F1-score of 0.8809, along with strong precision (0.8949) and recall (0.8672). The significantly lower test loss further indicates the robust generalization and training stability.

Key Observations:

- Optimizer selection significantly impacted the model's speed and performance
- A dropout rate of 0.2 struck a balance between underfitting and overfitting
- Traditional models were faster to train but less accurate in handling the sentiments



```
Test Accuracy: 0.8818
Test Loss: 0.2816804051399231
313/313 ━━━━━━━━━━━━━━━━━━━━ 10s 31ms/step
Precision: 0.8949
Recall: 0.8672
F1 Score: 0.8809
Confusion Matrix:
[[4448  513]
 [ 669 4370]]
```

Fig 1.3 DL - 003 Test Metrics Report

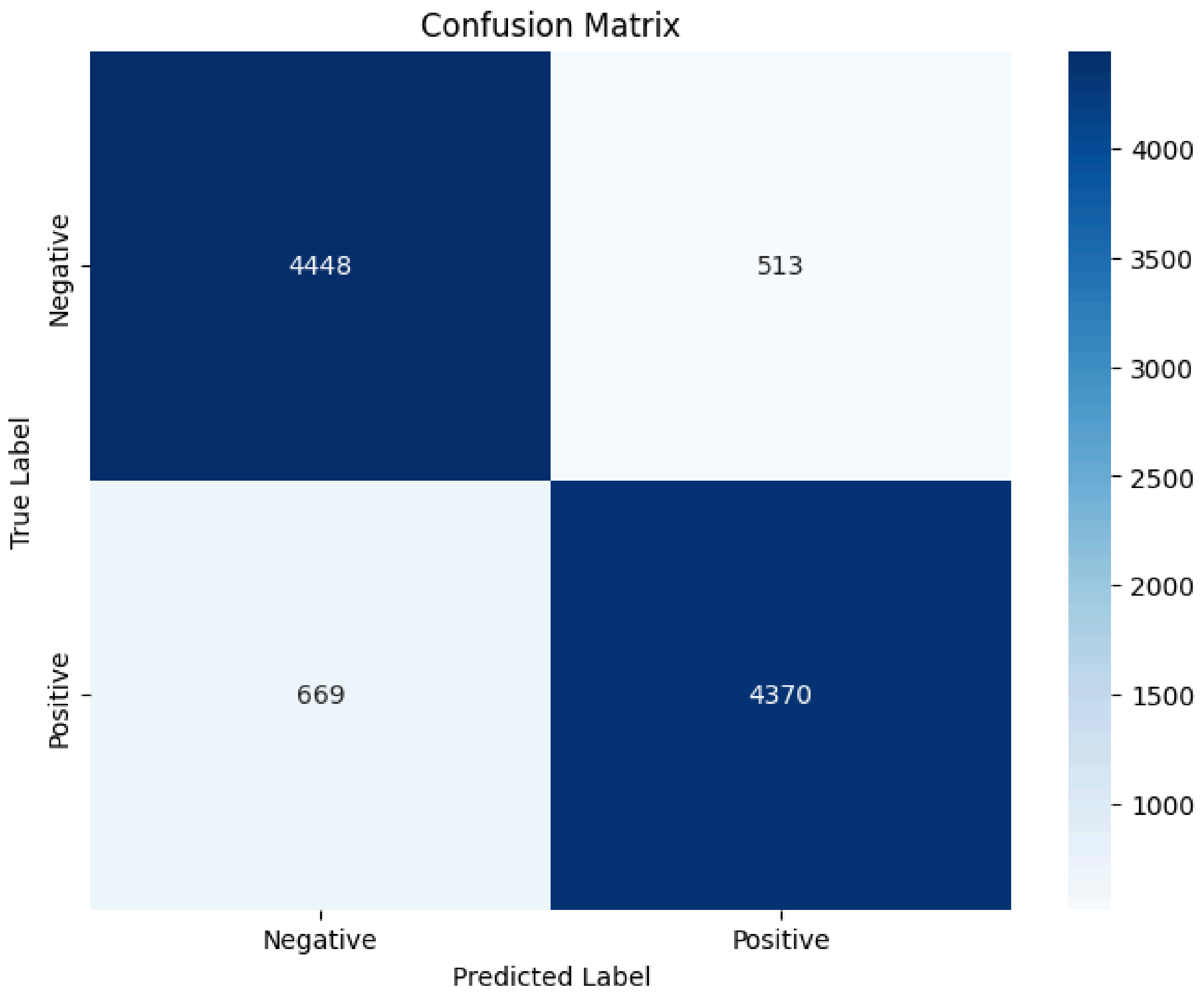


Fig 1.4 Confusion Matrix For DL - 003

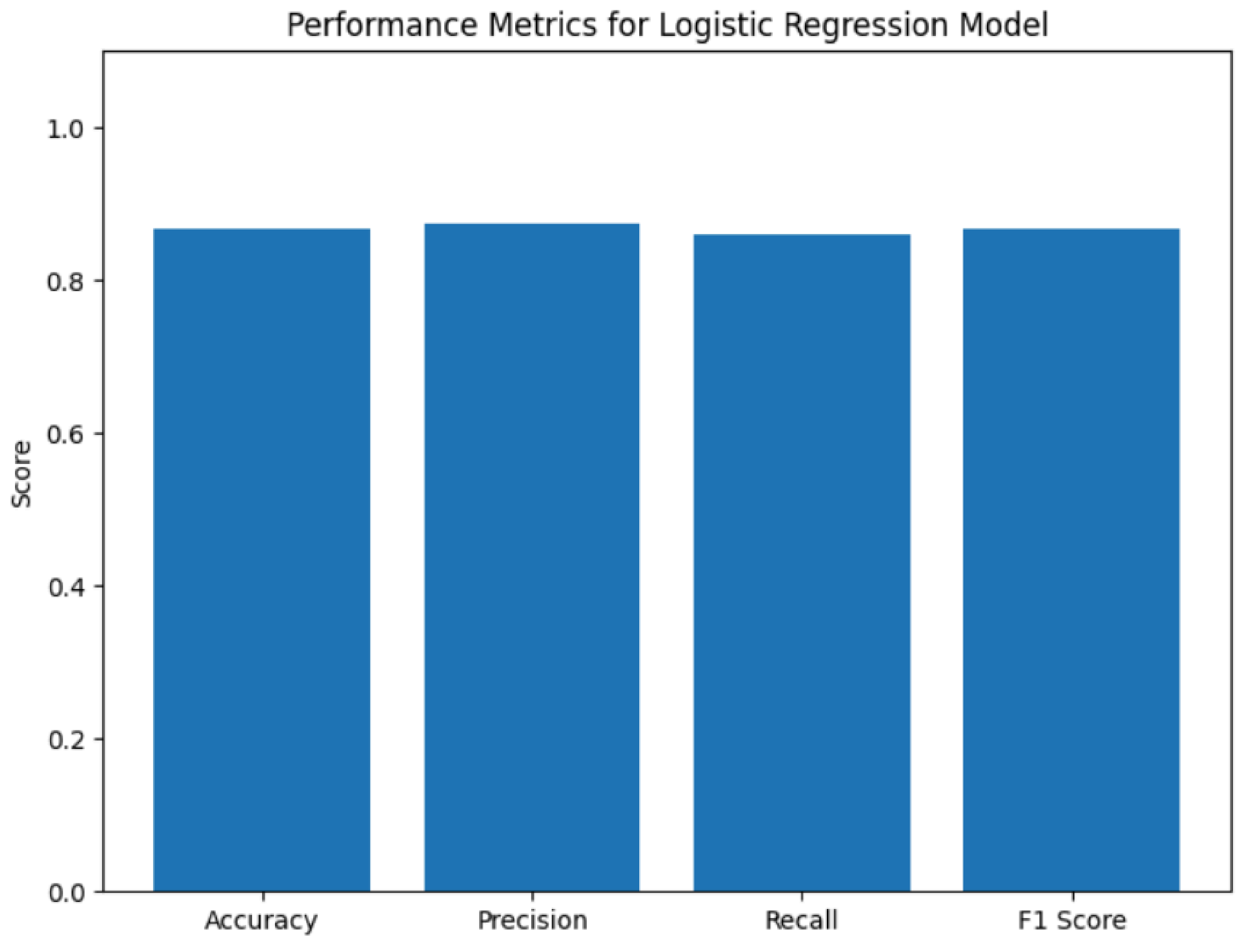


Fig 1.5 Logistic Regression Model Performance Metrics

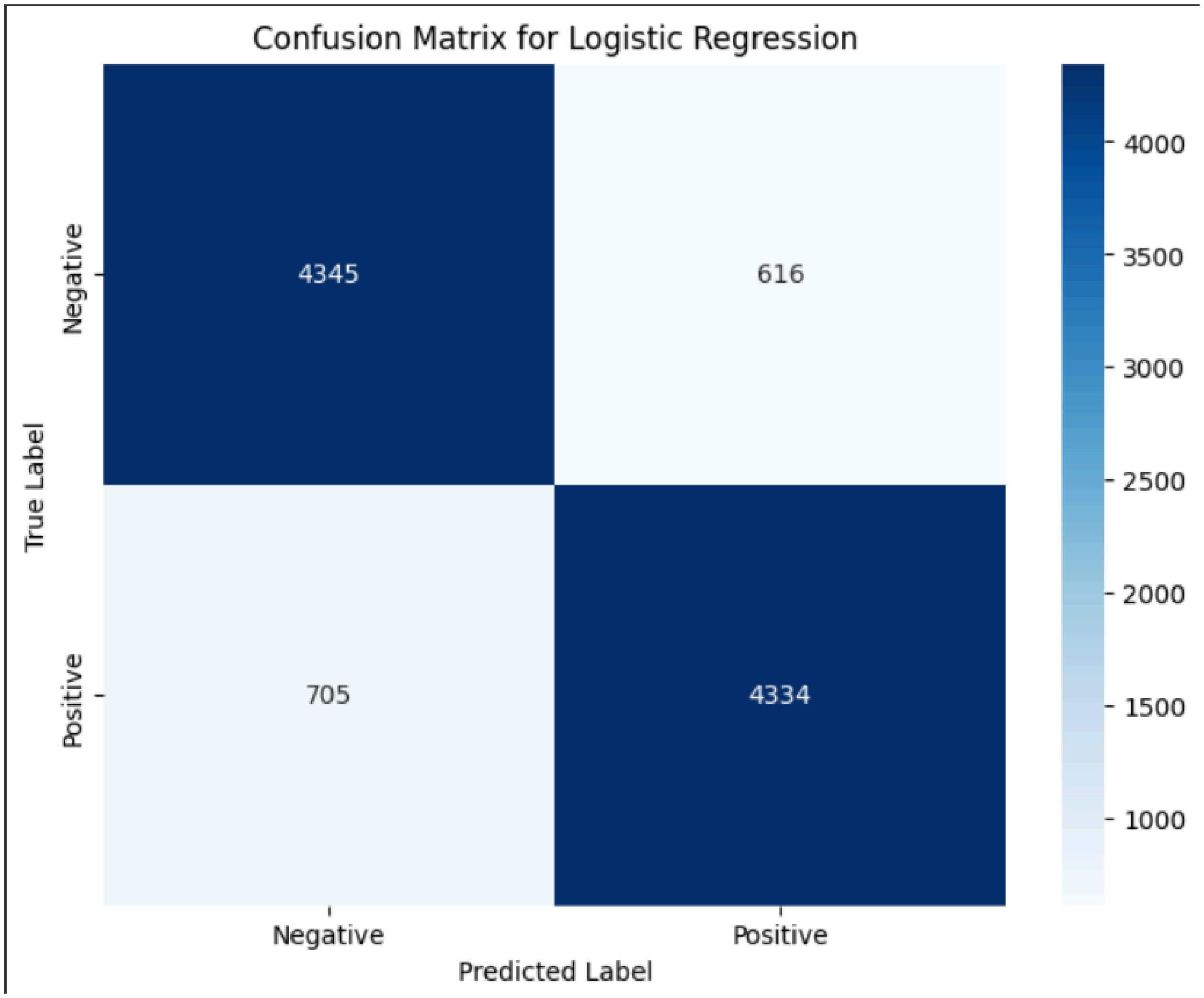


Fig 1.6 Logistic Regression Model Confusion Metrics

6. Conclusion

This project reaffirms the strength of deep learning in natural language processing tasks. While Logistic Regression is fast and reliable, deep learning models, when properly tuned, offer a richer understanding of context and sentiment.

The DL-003 model (Nadam optimizer with 0.4 dropout) demonstrated superior overall performance and is therefore recommended as the most reliable candidate for deployment in real sentiment analysis applications.

Future Recommendations

- Experiment with pretrained embeddings like GloVe or BERT
- Explore attention-based models for even deeper contextual insight
- Extend to multi-class sentiment datasets (e.g., neutral, sarcasm detection)

7. Team Contributions

- **Bernice**: Led data preparation and preprocessing pipeline design
- **Abubakar**: Built and evaluated the traditional ML model (Logistic Regression)
- **Purity**: Designed, implemented, and tuned the deep learning models
- **Peter**: Authored and edited the final academic report with guidance from team feedback

8. References

- [1] Y. Kim, "Convolutional Neural Networks for Sentence Classification," *EMNLP*, 2014.

  https://aclanthology.org/D14-1181.pdf

- [2] T. Mikolov et al., "Efficient Estimation of Word Representations in Vector Space," *arXiv preprint arXiv:1301.3781*, 2013.

  https://arxiv.org/abs/1301.3781

- [3] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.

  https://www.researchgate.net/publication/13853244_Long_Short-Term_Memory

- [4] IMDB 50K Sentiment Dataset:

  https://www.kaggle.com/code/jillanisofttech/imdb-movie-reviews-50k/input

- [5] TensorFlow and scikit-learn Documentation

  https://www.tensorflow.org/learn

- [6] "Opinion mining (sentiment mining)," TechTarget, [Online]. Available: https://www.techtarget.com/searchbusinessanalytics/definition/opinion-mining-sentiment-mining

- [7] K. O. Oluwaseun, "Comparative Study of Machine Learning Models for Sentiment Analysis: A Case of SVM, Naive Bayes and Logistic Regression," Medium, 30-Mar-2024. [Online]. Available: https://medium.com/@kisetzuu/comparative-study-of-machine-learning-models-for-sentiment-analysis-a-case-of-svm-naive-bayes-7311aea0f57f

- [8] M. Tareen et al., "Comparative Analysis of Word Embeddings for Sentiment Classification using Deep Learning," *Applied Sciences*, vol. 14, no. 24, p. 11519, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/14/24/11519