Text Classification System for Sentiment Analysis Report

# Abstract

This project aims to analyze the sentiment of text data using deep learning and traditional machine learning models. The dataset contains labeled text (positive, negative, neutral). After thorough preprocessing and tokenization, we trained multiple models including LSTM with pre-trained word embeddings, BERT (a transformer-based model), Logistic Regression, Support Vector Machine (SVM), and Random Forest. Our goal was to compare their performance and classify sentiment with high accuracy. We also implemented MLOps practices using MLflow and deployed the best model in a real-time application using Gradio and ngrok.

# Introduction

Sentiment Analysis is a key task in Natural Language Processing (NLP) that enables machines to understand human emotions and opinions from text data. It is widely used in customer feedback analysis, product reviews, and social media monitoring. This report presents the development and deployment of a robust sentiment analysis system using both machine learning and deep learning techniques..

# Literature Review

Numerous approaches have been proposed for sentiment analysis, ranging from traditional machine learning algorithms to modern transformer-based models. This project evaluates a combination of:

* Traditional ML models: Logistic Regression, SVM, Random Forest.
* Deep Learning: LSTM with GloVe embeddings.
* Transformer-based architecture: BERT (TFBertForSequenceClassification).

Each method offers trade-offs in terms of accuracy, training time, and complexity

# Methodology

**Dataset**

The dataset contains labeled English text samples categorized into positive, negative, and neutral sentiments. We combined both training and testing subsets from Kaggle to form a unified dataset.

**Preprocessing**

Steps included:

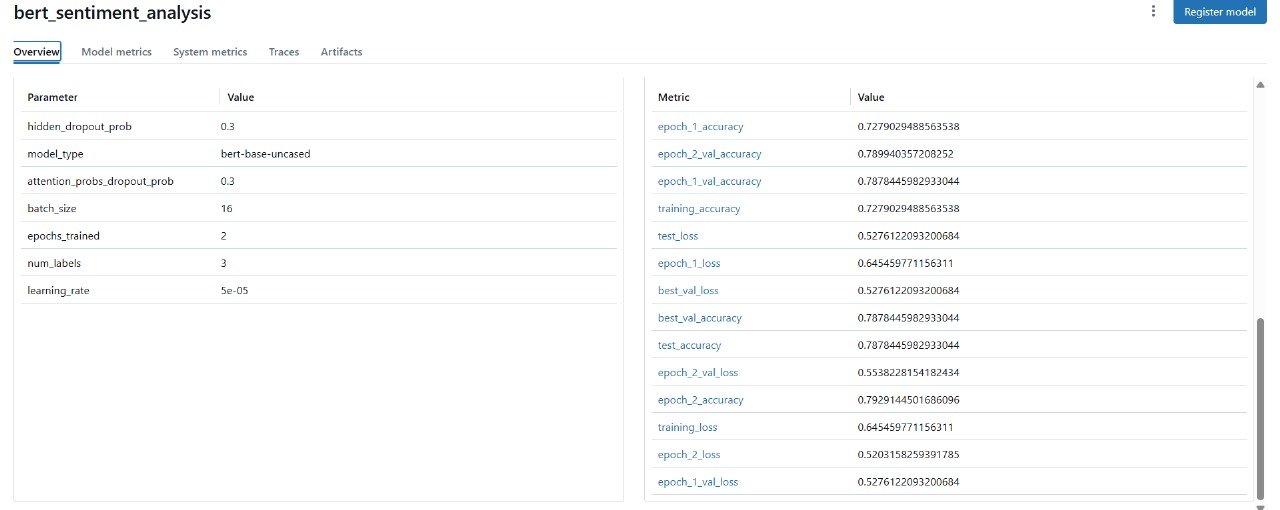
* Removing punctuation, special characters, and duplicates.
* Handling missing values.
* Text normalization (lowercasing, stemming, stopword removal).
* Tokenization using:
  + **Keras Tokenizer** (for LSTM)
  + **TF-IDF Vectorizer** (for traditional ML models)
  + **BERT Tokenizer** (bert-base-uncased for BERT model)

The data was split into training and testing sets, and sequences were padded for deep learning models.

# Implementation LSTM Model :

The model was built using TensorFlow/Keras. The architecture includes an trained and pre-trained (e.g., GloVe) embeddings layer, LSTM, Dropout, and Dense layers. Loss function used: sparse\_categorical\_crossentropy. Optimizer: Adam.

**Bert Model:**  
The model was built using TensorFlow/Keras. The architecture includes a **BertTokenizer** from the pre-trained **bert-base-uncased** model for tokenization. The model that we specifically used is **TFBertForSequenceClassification**, with Dropout layers for both hidden layers and attention probabilities. The optimizer used is **Adam**, and the loss function is **sparse\_categorical\_crossentropy**.



**Svm,Logistic Regression ,Random Forest Models:**

The model was built using a **Pipeline** in scikit-learn. The architecture includes a **TfidfVectorizer** . the solver specified as **liblinear**. This pipeline combines the text preprocessing (TF-IDF vectorization) with the logistic regression,Svm,Random Forest classifiers, for sentiment classification

# Results , Evaluation and Visualization

| **Model** | **Accuracy** | **Notes** |
| --- | --- | --- |
| **LSTM + GloVe** | **72%** | **Good generalization, overfitting avoided** |
| **BERT (TFBert)** | **79%** | **More powerful, slower to train** |
| **Logistic Regression** | **Best** | **Best accuracy among ML models** |
| **SVM** | **Moderate** | **Slightly lower accuracy** |
| **Random Forest** | **Moderate** | **High variance** |

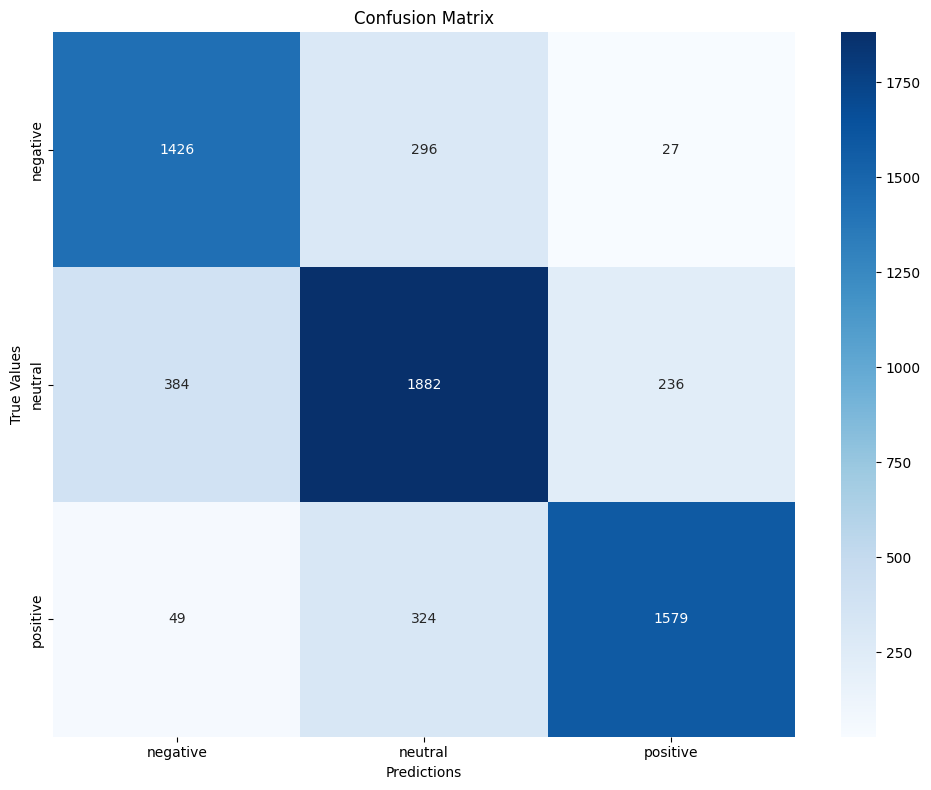
**For lstm model :**

The model achieved 72% on the test set. A confusion matrix and training curves were used to evaluate performance.

**For Bert model :**

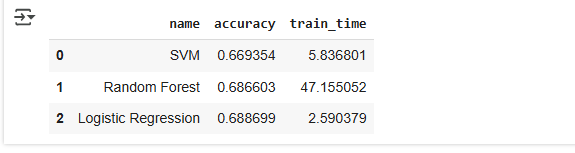
The model achieved 78% on the test set. A confusion matrix and training curves were used to evaluate performance.

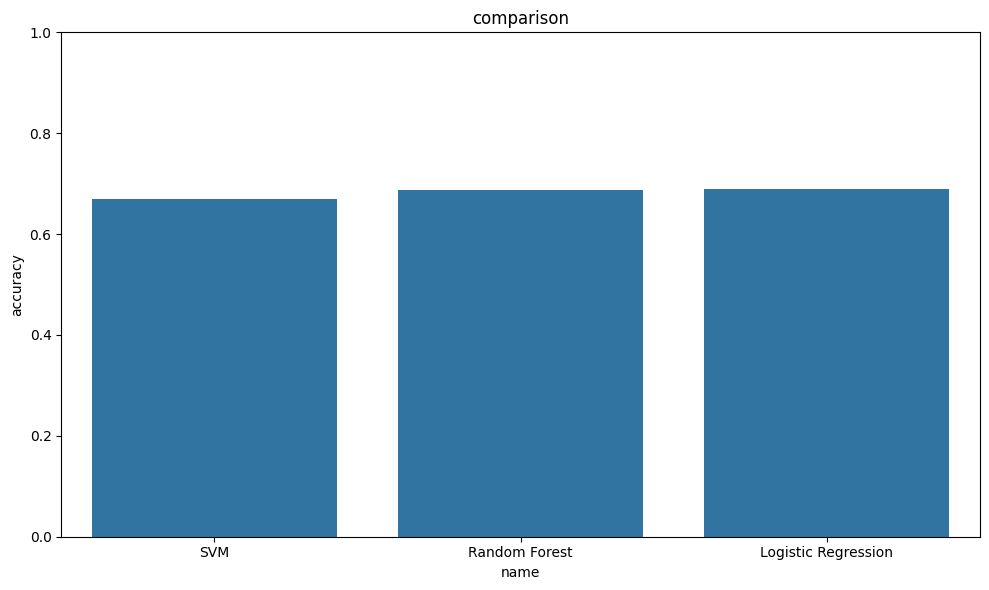
****



**For Svm,Logistic Regression ,Random Forest Models:**

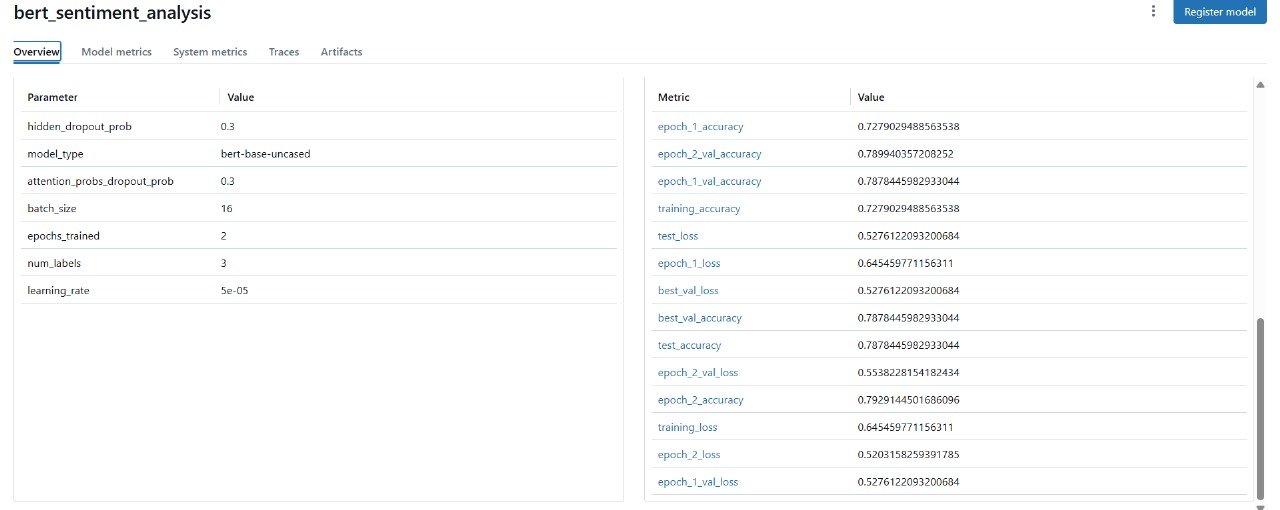
The best model in terms of accuracy **Logistic Regression**is **Logistic Regression**



 MLOps & Monitoring

We integrated MLOps practices to ensure reproducibility, tracking, and monitoring:

* **MLflow**:
  + Automatically logs training parameters, metrics, models, and artifacts.
  + Tracks multiple model experiments.
  + Facilitates model versioning and reproducibility.
* **For Bert model :**

****

# Deployment & Live Demo

We deployed the best model using:

* **Gradio**: To create a simple user interface for real-time prediction.
* **ngrok**: To expose the local server to the public internet.

The deployed system allows any user to input text and instantly receive sentiment predictions, making it an accessible demo of applied NLP.

# Business Impact

This sentiment analysis system has real-world applications in:

* **Customer Feedback Analysis**: Automatically classify product or service reviews to determine satisfaction.
* **Social Media Monitoring**: Track public opinion on brands or topics.
* **Spam Detection**: Identify negative or spammy messages.
* **News Categorization**: Classify headlines based on tone or subject.

This can help businesses automate decision-making, improve marketing strategies, and enhance user engagement

# Challenges

* BERT required large memory and long training times.
* Handling imbalanced classes and noisy data.
* Seamless integration of MLflow and Gradio.
* Deployment stability with ngrok.

# Conclusion

We successfully developed a robust sentiment analysis system that compares multiple models, integrates MLOps tools, and deploys a live, interactive demo. The project showcases the end-to-end application of modern NLP techniques and deployment in real-world settings