Project Title: Sentiment Analysis of Yelp Reviews Using Machine Learning and Deep Learning Models

1. Introduction

Sentiment analysis is a vital application of natural language processing (NLP) that involves classifying text as positive, negative, or neutral. This project aims to compare the performance of traditional machine learning algorithms with advanced deep learning models for sentiment analysis using the **Yelp Reviews dataset**.

2. Objective

- Use various machine learning algorithms (Logistic Regression, Decision Tree, Random Forest, SVM, KNN, Naive Bayes, GBM) and deep learning models (LSTM, BERT, and CNN) to classify sentiment in Yelp reviews.
- Evaluate and compare the accuracy, F1-score, and computational efficiency of these models.

3. Dataset: Yelp Reviews

- **Description**: The Yelp Reviews dataset contains customer reviews, including text and star ratings (1-5 stars).
- **Goal**: Classify the sentiment as either positive (4-5 stars) or negative (1-2 stars). You may discard neutral reviews (3 stars) for simplicity or include them for a multi-class classification task.

Link to Dataset: Yelp Reviews Dataset

4. Project Workflow

1. **Data Preprocessing**:
   - **Text Cleaning**: Remove special characters, convert to lowercase, remove stop words, and apply stemming or lemmatization.
   - **Tokenization**: Convert text into tokens for model input.
   - **Feature Extraction**:
     - **TF-IDF (Term Frequency-Inverse Document Frequency)** for machine learning models.
     - **Word Embeddings** (e.g., GloVe, Word2Vec) or **Tokenization** for deep learning models.
2. **Machine Learning Algorithms**:
   - **Logistic Regression**: A simple linear model used for binary classification.
   - **Decision Tree**: A tree-based model that splits data based on features.
   - **Random Forest**: An ensemble of decision trees.
   - **SVM (Support Vector Machine)**: A classification algorithm that separates classes with a hyperplane.
   - **KNN (K-Nearest Neighbors)**: A distance-based classification algorithm.
   - **Naive Bayes**: A probabilistic model based on Bayes' theorem.
   - **GBM (Gradient Boosting Machine)**: An ensemble technique to improve prediction performance.
3. **Deep Learning Models**:
   - **LSTM (Long Short-Term Memory)**:
     - Recurrent neural network architecture that captures sequential dependencies in the text.
     - Preprocessing: Convert reviews into sequences using word embeddings and pad them to the same length.
   - **BERT (Bidirectional Encoder Representations from Transformers)**:
     - Pretrained Transformer-based model designed for natural language understanding tasks.
     - Fine-tune BERT for sentiment classification by tokenizing the reviews and training on labeled data.
   - **CNN for Text**:
     - Convolutional Neural Networks (CNNs) can also be applied to text classification by using convolutions on word embeddings to capture important local features.
     - Preprocessing: Use word embeddings and feed them into a CNN model to extract features and classify the sentiment.
4. **Advanced Feature Engineering**:
   - **Text-based Features**: Extract additional features such as word counts, sentiment scores, or specific domain knowledge related to Yelp reviews.
   - **Dimensionality Reduction**: Apply techniques like **Principal Component Analysis (PCA)** to reduce the feature space for machine learning models, which can enhance model performance by removing noise.
   - **Domain-Specific Embeddings**: Utilize **pretrained embeddings** that are specific to the Yelp review domain, which could improve deep learning model performance by providing more context-relevant word representations.
5. **Training and Validation**:
   - **Training-Validation Split**: Split the data into training (80%) and validation/test (20%) sets.
   - **Cross-Validation**: For machine learning models, apply k-fold cross-validation to tune hyperparameters and prevent overfitting.
   - **Early Stopping**: Use early stopping for deep learning models to avoid overfitting during training.
6. **Evaluation Metrics**:
   - **Accuracy**: Percentage of correct predictions.
   - **Precision, Recall, F1-Score**: Metrics for model performance, especially important for imbalanced datasets.
   - **Confusion Matrix**: To visualize the model's classification performance.
   - **ROC Curve and AUC**: To compare classifier performance.
7. **Comparison of Models**:
   - **Performance**: Compare accuracy, precision, recall, and F1-score of machine learning and deep learning models.
   - **Computational Complexity**: Measure the training time and resource consumption of both approaches.

5. Implementation

1. **Machine Learning Models**:
   - Libraries: scikit-learn, pandas, numpy, matplotlib.
   - Steps: Preprocess data using TF-IDF, train and evaluate Logistic Regression, Decision Tree, Random Forest, SVM, KNN, Naive Bayes, and GBM.
2. **Deep Learning Models**:
   - Libraries: TensorFlow or PyTorch, transformers (for BERT).
   - Steps: Use word embeddings for LSTM and CNN, fine-tune BERT, and train both models. Validate and compare performance.

6. Tools & Environment

- **Development Platform**: Google Colab
- **Programming Language**: Python.
- **Libraries**:
  - For machine learning: scikit-learn, pandas, numpy.
  - For deep learning: TensorFlow, PyTorch, transformers (for BERT).

7. Conclusion

- Discuss the results, including which model performed better and why.
- Explain the trade-offs between machine learning and deep learning in terms of accuracy and computational efficiency.
- Highlight how deep learning (especially BERT) excels in capturing complex language nuances, but simpler machine learning models can be faster and more efficient for smaller datasets.