**Submission by:**

**Chirag Arora**

**102065031**

**GITHUB LINK FOR THIS PROJECT IS :** [**https://github.com/Regan17/Computer-Vision-Vehicle-classification-task**](https://github.com/Regan17/Computer-Vision-Vehicle-classification-task)

**Summary of Results:**

1. **Naive Bayes Classifier:**
   * Achieved an accuracy of 0.6541125
   * The confusion matrix provides insights into the performance of the model across different sentiment classes.
2. **Random Forest Classifier:**
   * Achieved an accuracy of 0.7821649.
   * The confusion matrix offers a detailed view of the classifier's performance.
3. **Bidirectional LSTM Model:**
   * Achieved an accuracy of 0.7481472
   * The model benefits from bidirectional LSTM layers and pre-trained GloVe embeddings, capturing complex relationships in the text.
4. **XGBoost Model:**
   * Achieved an accuracy of 0.7809500
   * The model leverages TF-IDF vectorization for feature representation.
5. **Dense Neural Network:**
   * Achieved an accuracy of 0.7933422
   * Utilizes a pipeline with CountVectorizer for text data, providing an alternative approach to neural network-based models.
6. **Support Vector Machine (SVM):**
   * Achieved an accuracy of 0.8036690
   * SVMs with TF-IDF vectorization showcase their effectiveness in text classification.

**Logic and Rationale for the Solution:**

1. **Choice of Models:**
   * Naive Bayes and Random Forest are classical models suitable for text classification tasks.
   * Bidirectional LSTM captures sequential dependencies in the text data.
   * XGBoost is a robust gradient boosting algorithm often effective in diverse scenarios.
   * Dense Neural Network and Support Vector Machine offer alternatives with different feature extraction approaches.
2. **Preprocessing:**
   * Text data is tokenized and padded for consistent input size.
   * GloVe embeddings enhance the model's understanding of word semantics.
   * Label encoding ensures numerical representation of sentiment labels.
3. **Evaluation Metrics:**
   * Accuracy provides a general overview of model performance.
   * Confusion matrices offer detailed insights into the true positives, true negatives, false positives, and false negatives for each sentiment class.

**Improvements with More Time:**

The most important improvement can be made to increase the accuracy above 0.85 is to use **transformer-based** models (e.g., BERT, GPT).I tried using the BERT model but colab’s GPU limitation and time that it takes made it impractical for me to make it happen but definitely it can help a lot in increasing the accuracy. I can provide the model’s code if you want.

Below are some other improvements that can be made:

1. **Hyperparameter Tuning:**
   * Fine-tune hyperparameters for each model to potentially improve performance.
   * Optimize tokenization parameters and sequence lengths for LSTM.
2. **Ensemble Methods:**
   * Explore ensemble methods to combine the strengths of multiple models for enhanced performance.
3. **Advanced Embeddings:**
   * Experiment with more advanced word embeddings or embeddings tailored for sentiment analysis tasks.
4. **Feature Engineering:**
   * Extract additional features from the text data or explore advanced feature engineering techniques.
5. **Cross-Validation:**
   * Implement cross-validation to obtain more robust and reliable performance metrics.
6. **Error Analysis:**
   * Conduct in-depth error analysis to understand the specific challenges each model faces and iteratively improve the solution.
7. **Deployment Considerations:**
   * Explore options for deploying the selected model to a production environment for real-world use.

Given more time, these steps could contribute to further refining and enhancing the sentiment analysis solution. The consideration of transformer models highlights their potential but also acknowledges the practical challenges associated with their implementation.