**Sentiment Analysis: Impact of Balanced vs. Unbalanced Data on Model Performance**

TEAM MEMBERS:

Urvisha Vora 101544453

Madhavi Raval 101579381

Namira Ganam 101561548

Srivatsan Rangarajan 101571286

Tarunsai Chukkala 101543772

Rutul Patel 101539636

Mansi Kharb 101562748

**TABLE OF CONTENT**

1. Background and Problem Statement ……………………………………………………………………..………1
   1. Overview ……………………………………………..…………………………………………………………….1
   2. Problem Statement ………………………………………………………………………….………………..1
2. Plan of Attack ……………………………………………………………………………………………..………………...1
   1. Approach Overview ……………………………………………………..…………………………………….1
3. The Database …………………………………………………………………………………..…………………………...2
   1. Data Source ……………………………………………………………………………..………………………..2
   2. Data Card …………………………………………………………………………………………………………..2
   3. Data Preprocessing …………………………………………………………………………………………….2
4. The Model ………………………………………………………………………………………………….……………….…3
   1. Benchmark Model: Logistic Regression ……………………………………….……………………. 3
   2. LSTM Model ……………………………………………………………………………………………………… 4
   3. Fine-tuned LSTM Model …………………………………………………………………………………… 5
5. Results …………………………………………………………………………………………………………………………..6
   1. Model Evaluation ……………………………………………………………………………………………….6
      1. Logistic Regression Model ………………………………………………………..…………… 7
      2. LSTM Model ………………………………………………………………………………….……… 7
      3. Fine-Tuned LSTM Model ……………………………………………………………..………... 8
   2. Visualizations ……………………………………………………………………………………………….….. 8
      1. Loss and Accuracy Curve for LSTM model (Balanced vs Unbalanced) ……………………………………………………………………………………………………………… 8
      2. Loss and Accuracy Curve for fine-tuned LSTM model (Balanced vs Unbalanced) ……………………………………………………………………………………….. 10
      3. Comparison of LSTM and Fine-Tuned LSTM Models ……………………….…… 11
6. Conclusions ……………………………………………………………………………………………………………….. 11
7. **Background and Problem Statement**

**1.1 Overview**

Sentiment analysis is a crucial task in natural language processing (NLP) that involves determining the sentiment behind textual data. In this project, we analyze how data imbalance affects model performance by training models on both balanced and unbalanced datasets. We explore the trade-offs between data balancing and model generalization using different machine learning and deep learning techniques.

**1.2 Problem Statement**

Many real-world sentiment analysis datasets suffer from class imbalance, where certain sentiment classes (e.g., positive reviews) dominate others. This imbalance can lead to biased models that perform well on majority classes but poorly on minority ones. The objective of this project is to:

* Investigate how data imbalance affects model performance.
* Compare a LSTM Model with fine-tuned LSTM deep learning models.
* Analyze how balancing the dataset influences training dynamics and generalization.

**2. Plan of Attack**

**2.1 Approach Overview**

To systematically study the impact of data balancing, we adopted the following approach:

* Dataset Exploration & Preprocessing – Investigated class distribution and performed text preprocessing.
* Baseline Model (Logistic Regression) – Established a benchmark model for comparison.
* Deep Learning Models
  + LSTM Model trained on the original unbalanced dataset.
  + Fine-tuned LSTM Model trained on a balanced dataset.
* Evaluation Metrics – Compared models using accuracy, confusion matrices, and Accuracy-Loss curve.

**3. The Database**

**3.1 Data Source**

The dataset used is the Coursera Course Reviews dataset from Kaggle ([link](https://www.kaggle.com/datasets/septa97/100k-courseras-course-reviews-dataset)). It contains user reviews along with sentiment labels.

**3.2 Data Card**

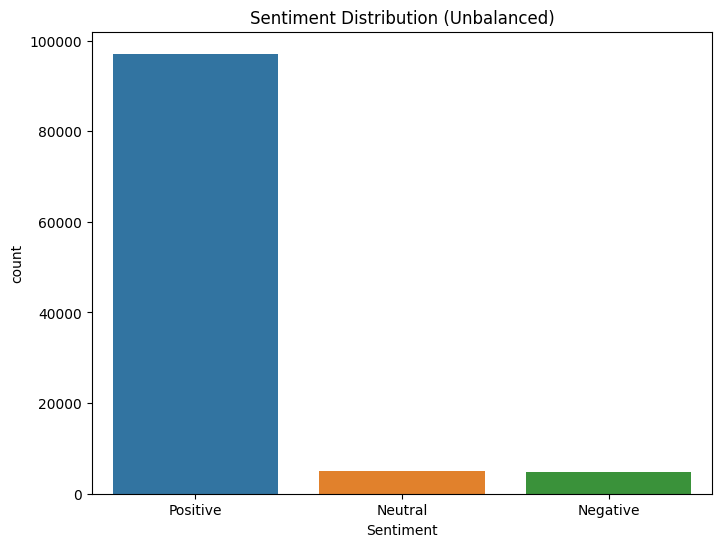
* Total Entries: 140320
* Total Columns: 3, Id (The unique identifier for a review), Review (The actual course review), Label (The rating of the course review)
* Label consist of rating from 1 to 5.

| Label count | |
| --- | --- |
| 5 | **106514** |
| 4 | **22460** |
| 3 | **5923** |
| 2 | **2554** |
| 1 | **2866** |

**3.3 Data Preprocessing**

The following preprocessing steps were performed:

* Text Cleaning: Removed stopwords, punctuation, and special characters.
* Tokenization & Padding: Tokenized text and applied padding for deep learning models.
* Class Balancing: Created a balanced dataset by oversampling the majority class.
* Train-Test Split: Split data into training, validation, and test sets.
* Fixing Gibbersh: Some gibberish text was removed by applying Unidecode.
* Categorizing Labels: Splitting labels into 3 parts, Positive (Label = 1, 2), Negative (Label = 3), and neutral (Label = 4,5). The visualization of the Categorized label is as follows, which clearly addresses the imbalance in the dataset.



**4. The Model**

**4.1 Benchmark Model: Logistic Regression**

As a baseline, we trained a simple logistic regression model using TF-IDF features. This model provides an initial benchmark to compare deep learning approaches.

**Code Snippets:**

**encoder = LabelEncoder()**

**y\_train = encoder.fit\_transform(y\_train)**

**y\_test = encoder.transform(y\_test)**

**tfidf = TfidfVectorizer(max\_features=5000)**

**X\_train\_tfidf = tfidf.fit\_transform(X\_train)**

**X\_test\_tfidf = tfidf.transform(X\_test)**

**log\_model = LogisticRegression()**

**log\_model.fit(X\_train\_tfidf, y\_train)**

**y\_pred\_log = log\_model.predict(X\_test\_tfidf)**

The logistic regression model serves as a simple baseline to compare against more complex deep learning approaches. It uses TF-IDF vectorization to transform text into numerical representations before training a logistic regression classifier.

**4.2 LSTM Model**

**LSTM Model**

* A sequential Long Short-Term Memory (LSTM) network was implemented to capture long-range dependencies in text.
* Trained on the original, unbalanced dataset and balanced dataset.
* Optimization used Adam with categorical cross-entropy loss.

**Code snippits:**

**tokenizer = Tokenizer(num\_words=5000)**

**tokenizer.fit\_on\_texts(X\_train)**

**X\_train\_seq = tokenizer.texts\_to\_sequences(X\_train)**

**X\_test\_seq = tokenizer.texts\_to\_sequences(X\_test)**

**X\_train\_pad = pad\_sequences(X\_train\_seq, maxlen=100)**

**X\_test\_pad = pad\_sequences(X\_test\_seq, maxlen=100)**

**unbal\_lstm = Sequential([**

**Embedding(input\_dim=5000, output\_dim=128, input\_length=100),**

**LSTM(128, dropout=0.5, recurrent\_dropout=0.5),**

**Dense(3, activation='softmax')**

**])**

**unbal\_lstm.compile(loss='sparse\_categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])**

**early\_stopping = EarlyStopping(monitor='val\_loss', patience=3, restore\_best\_weights=True)**

**history\_unbalanced\_lstm=unbal\_lstm.fit(X\_train\_pad, y\_train, epochs=10, batch\_size=64, validation\_data=(X\_test\_pad, y\_test), callbacks=[early\_stopping])**

A basic LSTM model was implemented to capture sequential dependencies in text. It consists of an embedding layer, a single LSTM layer with dropout, and a softmax output layer for classification.

**4.3 Fine-tuned LSTM Model**

* Trained on both balanced dataset and unbalanced dataset.
* Hyperparameter tuning involved:
  + Varying the number of LSTM layers and units.
  + Adjusting dropout rates to prevent overfitting.
  + Experimenting with learning rates.
* The expectation was that balancing the dataset would improve generalization.

**max\_words = 20000 # Vocabulary size**

**max\_len = 100 # Max length of sequences**

**tokenizer = Tokenizer(num\_words=max\_words)**

**tokenizer.fit\_on\_texts(english\_reviews\_fLstm\_df['cleaned\_review'])**

**sequences = tokenizer.texts\_to\_sequences(english\_reviews\_fLstm\_df['cleaned\_review'])**

**X = pad\_sequences(sequences, maxlen=max\_len)**

**y = english\_reviews\_fLstm\_df['Label'].values**

**# Split into training and testing sets**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42, stratify=y)**

**# Compute class weights**

**class\_weights = compute\_class\_weight(class\_weight="balanced", classes=np.unique(y\_train), y=y\_train)**

**class\_weights\_dict = {i: class\_weights[i] for i in range(len(class\_weights))}**

**# Build LSTM model for unbalanced data**

**unbalanced\_lstm\_model = Sequential([**

**Embedding(max\_words, 128, input\_length=max\_len),**

**Bidirectional(LSTM(64, return\_sequences=True)),**

**Dropout(0.5),**

**Bidirectional(LSTM(32)),**

**Dense(64, activation='relu'),**

**Dropout(0.5),**

**Dense(1, activation='sigmoid')**

**])**

**unbalanced\_lstm\_model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])**

**# unbalanced\_lstm\_model.summary()**

**early\_stopping = EarlyStopping(monitor='val\_loss', patience=3, restore\_best\_weights=True)**

**# Train the model on unbalanced data**

**history\_unbalanced\_ft = unbalanced\_lstm\_model.fit(**

**X\_train, y\_train,**

**validation\_data=(X\_test, y\_test),**

**epochs=10,**

**batch\_size=64,**

**verbose=1,**

**class\_weight=class\_weights\_dict,**

**callbacks=[early\_stopping]**

**)**

The fine-tuned LSTM model includes bidirectional LSTMs and additional dropout layers to enhance generalization and prevent overfitting. It also incorporates class weighting to compensate for any remaining imbalance.

**5. Results**

**5.1 Model Evaluation**

The performance of each model was evaluated using accuracy, precision, recall, and f1-score. The results provide significant insights into how different models behave when trained on balanced and unbalanced datasets.

**Model Performance Comparison Table:**

| **Model** | **Dataset Type** | **Accuracy** | **Precision (Avg)** | **Recall (Avg)** | **F1-score (Avg)** |
| --- | --- | --- | --- | --- | --- |
| **Logistic Regression** | **Unbalanced** | **0.9278** | **0.91** | **0.93** | **0.91** |
| **Logistic Regression** | **Balanced** | **0.8349** | **0.92** | **0.83** | **0.87** |
| **LSTM** | **Unbalanced** | **0.9306** | **0.91** | **0.93** | **0.92** |
| **LSTM** | **Balanced** | **0.8546** | **0.92** | **0.85** | **0.88** |
| **Fine-Tuned LSTM** | **Unbalanced** | **0.91** | **0.94** | **0.91** | **0.92** |
| **Fine-Tuned LSTM** | **Balanced** | **0.93** | **0.93** | **0.93** | **0.93** |

**5.1.1 Logistic Regression Model**

Unbalanced Dataset:

* Accuracy: 0.9278
* Class 2 (majority class) has very high precision (0.94) and recall (0.99), dominating the overall performance.
* Minority classes (0 and 1) have low recall, indicating poor prediction of less frequent sentiments.

Balanced Dataset:

* Accuracy: 0.8349 (lower than unbalanced model but more fair across classes).
* Minority class recall improves significantly.
* Precision of class 2 drops slightly, leading to a better-balanced performance.

Key Observation: The balanced dataset improves recall for minority classes at the cost of overall accuracy.

**5.1.2 LSTM Model**

Unbalanced Dataset:

* Accuracy: 0.9306 (slightly better than logistic regression).
* Class 2 maintains high recall (0.99) but classes 0 and 1 remain underrepresented.

Balanced Dataset:

* Accuracy: 0.8546.
* Better recall for minority classes, similar to logistic regression but with stronger f1-scores.

Key Observation: LSTM improves overall representation but still struggles with minority class predictions.

**5.1.3 Fine-Tuned LSTM Model**

Unbalanced Dataset:

* Accuracy: 0.91.
* Class 0 recall significantly improved (0.82 vs previous models ~0.50-0.60).
* Better balance in predictions while maintaining strong precision.

Balanced Dataset:

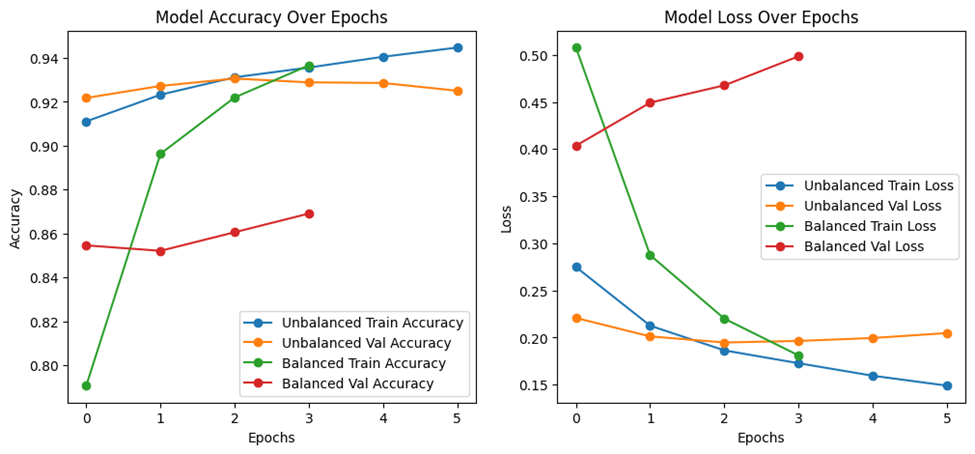
* Accuracy: 0.93 (best-performing model in terms of balance).
* Class 0 recall reaches 0.67.
* F1-scores show an overall well-balanced classification.

Key Insights:

* Fine-tuned LSTM significantly enhances minority class predictions.
* The balanced dataset further improves the recall of underrepresented classes without sacrificing overall performance.
* The choice of dataset balance depends on whether overall accuracy or fairness across classes is the priority.

**5.2 Visualizations**

**5.2.1 Loss and Accuracy Curve for LSTM model (Balanced vs Unbalanced)**

****

Model Accuracy Over Epochs:

* Unbalanced Train Accuracy (Blue Line): Starts around 91% and steadily improves to 94%.
* Unbalanced Validation Accuracy (Orange Line): Starts slightly higher and remains stable around 92%, with a slight drop at the end.
* Balanced Train Accuracy (Green Line): Starts lower (~80%) but rapidly increases, catching up with the unbalanced model.
* Balanced Validation Accuracy (Red Line): Gradually improves but remains slightly below the unbalanced validation accuracy.

Key Observation:

* The balanced model starts off weaker but quickly catches up, indicating better generalization.
* The unbalanced model performs consistently well on validation, but this could mean it's overfitting to specific patterns in the dataset.

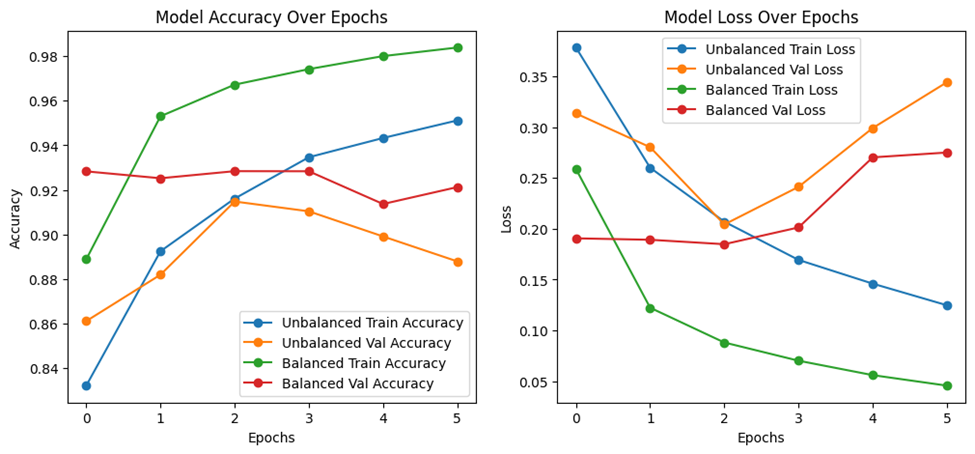
Model Loss Over Epochs:

* Unbalanced Train Loss (Blue Line): Decreases steadily, as expected.
* Unbalanced Validation Loss (Orange Line): Decreases initially but starts increasing slightly, indicating possible overfitting.
* Balanced Train Loss (Green Line): Starts high (~0.5) but drops significantly, showing the model is learning fast.
* Balanced Validation Loss (Red Line): Increases, which might indicate underfitting or noisy validation data.

Key Conclusion:

* The balanced model learns significantly in early epochs but still struggles with validation performance.
* The unbalanced model has lower validation loss, but its increasing trend suggests it might overfit.

**5.2.2 Loss and Accuracy Curve for fine-tuned LSTM model (Balanced vs Unbalanced)**

****

#### Loss Trends for Fine-Tuned LSTM:

* Unbalanced Train Loss (Blue Line): Decreases steadily, showing the model learns effectively.
* Unbalanced Validation Loss (Orange Line): Decreases at first but then increases, indicating overfitting.
* Balanced Train Loss (Green Line): Starts low and continues decreasing, meaning the model is learning efficiently.
* Balanced Validation Loss (Red Line): Remains mostly stable but slightly increases.

Key Takeaways:

* Balancing the dataset significantly improves model performance.
  + The Balanced model (Green and Red lines) consistently outperforms the Unbalanced model (Blue and Orange lines) in both accuracy and loss, indicating better generalization.
* Unbalanced data leads to overfitting.
  + The Unbalanced model shows an increasing training accuracy while validation accuracy plateaus or decreases, meaning it memorizes training data instead of learning generalizable patterns.
* Balanced data improves generalization.
  + The Balanced model shows consistent improvement in both training and validation accuracy, indicating better generalization and reduced overfitting.

**5.2.3 Comparison of LSTM and Fine-Tuned LSTM Models**

1. Training and Validation Accuracy:
   * The LSTM model showed strong performance, but the fine-tuned LSTM model improved generalization, especially on balanced data.
   * Fine-tuning led to better validation accuracy stability compared to the basic LSTM.
2. Overfitting Trends:
   * The LSTM model (unbalanced) exhibited clear overfitting, with validation loss increasing after initial improvements.
   * The fine-tuned LSTM reduced overfitting through bidirectional layers and dropout, improving balance in learning.
3. Effect of Data Balancing:
   * Both models benefited from balancing, but the fine-tuned LSTM model showed a more significant performance boost on balanced data, with validation accuracy closely following training accuracy.
   * This indicates better generalization and stability, avoiding biases seen in the unbalanced LSTM.

Key Takeaway:

Fine-tuning helped mitigate overfitting, and dataset balancing was more effective in the fine-tuned LSTM, leading to improved robustness in sentiment classification

**6. Conclusions**

This study highlights the significant impact of data imbalance on sentiment analysis models. The results show that balancing the dataset improves generalization, reducing overfitting and bias towards majority classes. The fine-tuned LSTM model, with additional layers and dropout regularization, further enhanced performance, particularly in handling minority classes more effectively.

#### **Did the plan of attack make sense?**

Yes, the systematic approach—starting with dataset exploration, implementing a baseline model, and then comparing different LSTM architectures—provided clear insights into the effects of data balancing. The evaluation using multiple metrics ensured a thorough assessment of model performance.

#### 

#### **What would we do differently next time?**

* Explore alternative balancing techniques: Instead of oversampling, we could experiment with cost-sensitive learning.
* Incorporate additional models: Testing transformer-based models like BERT could offer further insights.
* Hyperparameter tuning: More extensive tuning, including different optimizers and learning rate schedules, might further improve model performance.

#### **Final Takeaway**

Balancing the dataset significantly improves generalization and fairness across sentiment classes, while fine-tuning LSTM architectures helps mitigate overfitting. The choice between raw accuracy and balanced class performance depends on the specific application needs.