Sentiment Classification of Amazon Data

By: Charles Wibonele, Habeeb Ajibola,
Tochukwu Nto Mbah

Introduction

Objective

- The purpose of this presentation is to demonstrate the use of sentiment classification on Amazon product reviews
- We will also present comparisons and evaluations of the models used.

Key Points addressed

- Growing reliance on online reviews for product and business decision insights
- Vast volume of data makes manual analysis impractical
- Traditional machine-based tools may fail to capture nuanced emotions and context in text

Introduction pt.2

What is Sentiment Classification?

- The process of automatically labeling text according to its emotional tone, typically as negative, neutral, or positive.
- Frequently used for customer feedback, product reviews, and social media posts to gauge overall sentiment.

Why Use BERT?

- Context-aware deep learning model for NLP tasks, capturing subtle language nuances that simpler models may miss.
- Particularly adept at understanding nuanced expressions, making it highly effective for sentiment classification.

Raw Data

	id	asins	brand	categories	colors	dateAdded	dateUpdated	dimension	ean	keys	...	reviews.rating	reviews.sourceURLs	reviews.text	reviews.title	reviews.userCity	reviews.userProvince	reviews.username
0	AVpe7AsMilAPnD_xQ78G	B00QJDU3KY	Amazon	Amazon Devices,mazon.co.uk	NaN	2016-03-08T20:21:53Z	2017-07-18T23:52:58Z	169 mm x 117 mm x 9.1 mm	NaN	kindlepaperwhite/b00qjdu3ky	...	5.0	https://www.amazon.com/Kindle-Paperwhite-High-...	I initially had trouble deciding between the p...	Paperwhite voyage, no regrets!	NaN	NaN	Cristina M
1	AVpe7AsMilAPnD_xQ78G	B00QJDU3KY	Amazon	Amazon Devices,mazon.co.uk	NaN	2016-03-08T20:21:53Z	2017-07-18T23:52:58Z	169 mm x 117 mm x 9.1 mm	NaN	kindlepaperwhite/b00qjdu3ky	...	5.0	https://www.amazon.com/Kindle-Paperwhite-High-...	Allow me to preface this with a little history...	One Simply Could Not Ask For More	NaN	NaN	Ricky
2	AVpe7AsMilAPnD_xQ78G	B00QJDU3KY	Amazon	Amazon Devices,mazon.co.uk	NaN	2016-03-08T20:21:53Z	2017-07-18T23:52:58Z	169 mm x 117 mm x 9.1 mm	NaN	kindlepaperwhite/b00qjdu3ky	...	4.0	https://www.amazon.com/Kindle-Paperwhite-High-...	I am enjoying it so far. Great for reading. Ha...	Great for those that just want an e-reader	NaN	NaN	Tedd Gardiner
3	AVpe7AsMilAPnD_xQ78G	B00QJDU3KY	Amazon	Amazon Devices,mazon.co.uk	NaN	2016-03-08T20:21:53Z	2017-07-18T23:52:58Z	169 mm x 117 mm x 9.1 mm	NaN	kindlepaperwhite/b00qjdu3ky	...	5.0	https://www.amazon.com/Kindle-Paperwhite-High-...	I bought one of the first Paperwhites and have...	Love / Hate relationship	NaN	NaN	Dougal
4	AVpe7AsMilAPnD_xQ78G	B00QJDU3KY	Amazon	Amazon Devices,mazon.co.uk	NaN	2016-03-08T20:21:53Z	2017-07-18T23:52:58Z	169 mm x 117 mm x 9.1 mm	NaN	kindlepaperwhite/b00qjdu3ky	...	5.0	https://www.amazon.com/Kindle-Paperwhite-High-...	I have to say upfront - I don't like coroporat...	I LOVE IT	NaN	NaN	Miljan David Tanic

Dataset Overview

Dataset: Amazon Reviews

Columns Used:

- reviews.text: The review text
- reviews.rating: The star rating (1-5)
- sentiment: Derived from the rating (Negative, Neutral, Positive)
- reviews.date: Timestamp of the review

Data Cleaning Steps:

- Removed punctuation and HTML tags
- Converted text to lowercase
- Removed stopwords

Cleaned Review text data preview

4. Cleaning the Review Text

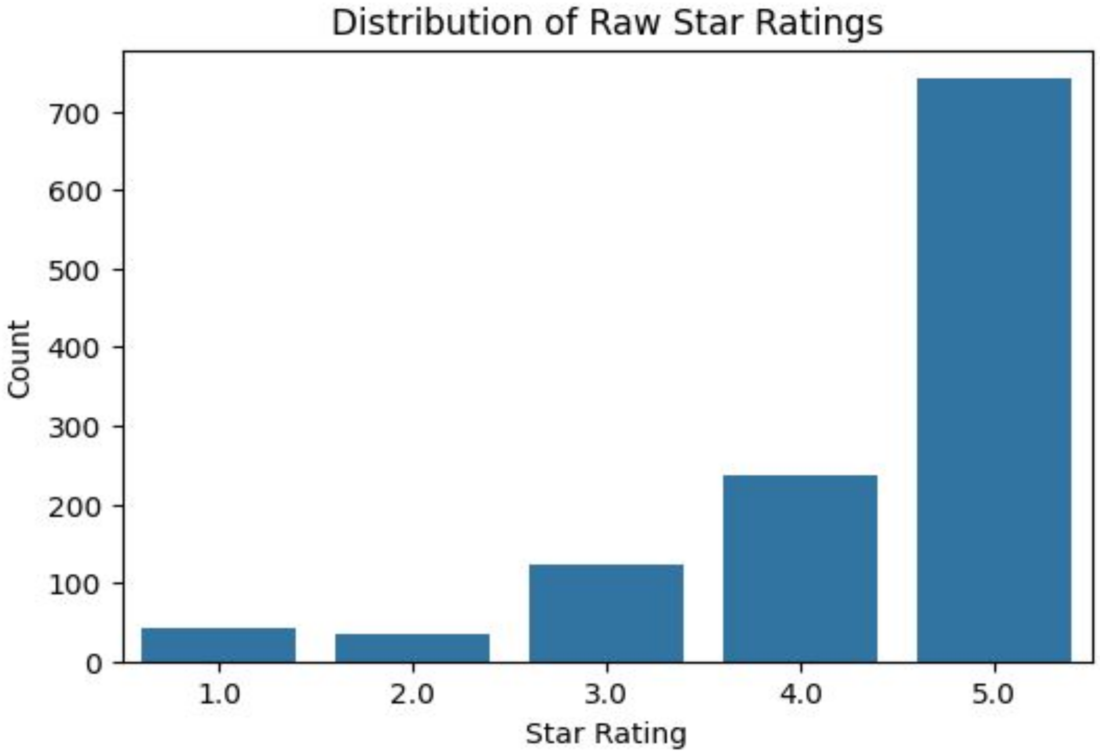
```
] def clean_text(text):
      text = re.sub(r'<.*?>', '', str(text))          # Remove HTML tags
      text = re.sub(r'[^\w\s]', '', text)             # Remove punctuation
      text = text.lower()                             # Lowercase
      stop_words = set(stopwords.words('english'))
      text = ' '.join(word for word in text.split()
                      if word not in stop_words)      # Remove stopwords
      return text

  df['cleaned_review'] = df['reviews.text'].apply(clean_text)
```

Original vs. Cleaned Review (first 5 rows):

	reviews.text	cleaned_review
0	I initially had trouble deciding between the p...	initially trouble deciding paperwhite voyage r...
1	Allow me to preface this with a little history...	allow preface little history casual reader own...
2	I am enjoying it so far. Great for reading. Ha...	enjoying far great reading original fire since...
3	I bought one of the first Paperwhites and have...	bought one first paperwhites pleased constant ...
4	I have to say upfront - I don't like coroporat...	say upfront dont like coroporate hermetically ...

Word cloud visual of ‘cleaned_review’ text column

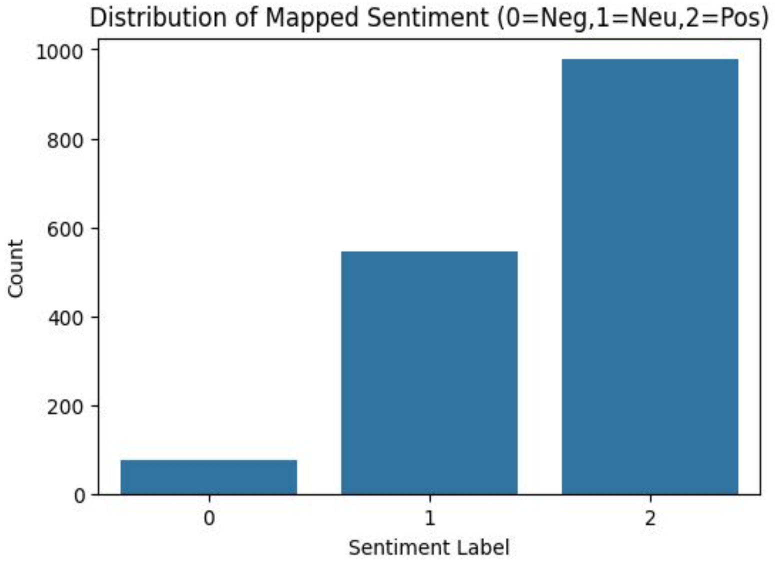
Bar graph visual of the raw ‘ratings’ column



Bar graph visual of new Sentiment labeling

6. Mapping `'rating'` column to Sentiment

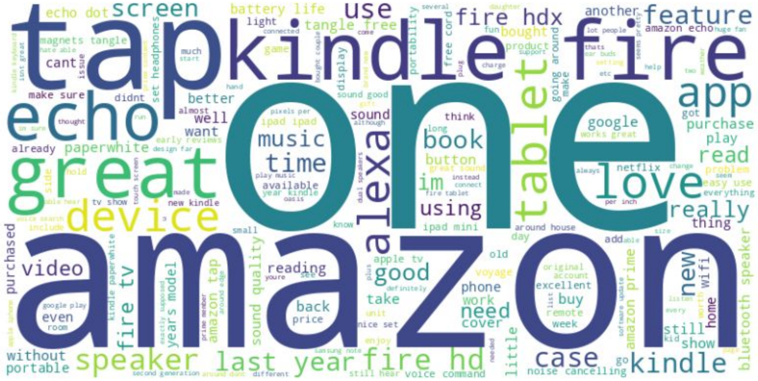
```python
def rating_to_sentiment(rating):
    # Convert rating to numeric
    rating = pd.to_numeric(rating, errors='coerce')
    if rating >= 4:
        return 2  # Positive
    elif rating == 3:
        return 1  # Neutral
    else:
        return 0  # Negative

df['reviews.rating'] = pd.to_numeric(df['reviews.rating'], errors='coerce')
df['reviews.rating'].fillna(3, inplace=True)  # default to neutral if missing
df['sentiment'] = df['reviews.rating'].apply(rating_to_sentiment)
```



Word Cloud for Negative Reviews

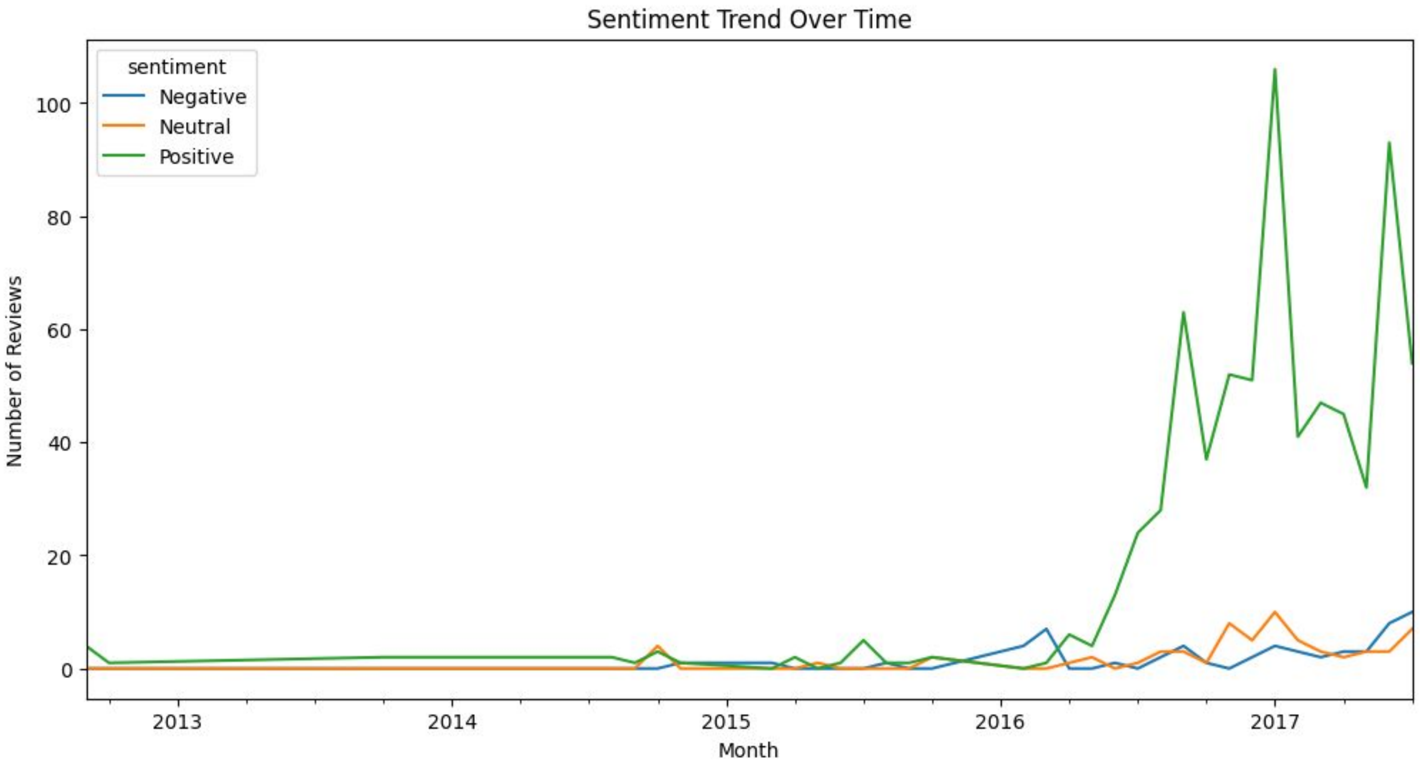
Word Cloud for Neutral Reviews

Word Cloud for Positive Reviews



Word Cloud Visual of Reviews Per Sentiment Category

Sentiment Trend over time (before class weights)



Traditional ML for Sentiment Classification on Amazon Reviews

Data Preprocessing & Feature Extraction:

- Uses **cleaned_review** & ***sentiment*** labels as columns
- Train-Test Split: 80/20
- Text Vectorization: CountVectorizer + TFIDFTransformer

Overall Performance:

- Both models achieve similar overall accuracy, indicating that traditional ML methods captures a poor general sentiment trend in the dataset.

Class Imbalance Effects:

- The negative class consistently shows much lower recall and F1-scores compared to neutral and positive classes.
- The performance drop for the negative class may lead to biased predictions favoring neutral or positive sentiments.

```
SVM Classifier Report:
              precision    recall  f1-score   support

           0       0.50      0.27      0.35        11
           1       0.60      0.69      0.64       107
           2       0.81      0.77      0.79       202

    accuracy                           0.72       320
   macro avg       0.64      0.58      0.60       320
weighted avg       0.73      0.72      0.73       320

Confusion Matrix for SVM:
[[  3   4   4]
 [  1  74  32]
 [  2  45 155]]
```

```
Random Forest Classifier Report:
              precision    recall  f1-score   support

           0       0.67      0.18      0.29        11
           1       0.64      0.64      0.64       107
           2       0.78      0.81      0.80       202

    accuracy                           0.73       320
   macro avg       0.69      0.54      0.57       320
weighted avg       0.73      0.73      0.72       320

Confusion Matrix for Random Forest:
[[  2   1   8]
 [  1  68  38]
 [  0  38 164]]
```

Pre-trained Bert Model Overview

Pretrained BERT (bert-base-uncased)

Fine-tuned for sentiment classification (3 classes):

- 0 = Negative
- 1 = Neutral
- 2 = Positive

Initial Training:

- Imbalanced dataset affecting performance
- Overfitting to dominant classes

Epoch	Training Loss	Validation Loss
1	No log	0.623180
2	No log	0.624887
3	No log	0.624916

```
TrainOutput(global_step=480, training_loss=0.571120707722982, metrics={'train_runtime': 202.7978, 'train_samples_per_second': 18.891, 'train_steps_per_second': 2.367, 'total_flos': 251996875828992.0, 'train_loss': 0.5711207707722982, 'epoch': 3.0})
```

Initial Model Performance (Before Class Weighting)

Accuracy: 71%

F1-Score: 70%

Confusion Matrix Observations:

- The model **over-predicts positive sentiment**.
- **Very poor recall for negative and neutral classes**.
- Imbalance in training data affects classification.

```
Accuracy: 0.71
Precision: 0.69
Recall: 0.71
F1-Score: 0.70

Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        11
           1       0.60      0.65      0.62       107
           2       0.78      0.78      0.78       202

    accuracy                           0.71       320
   macro avg       0.46      0.48      0.47       320
weighted avg       0.69      0.71      0.70       320
```

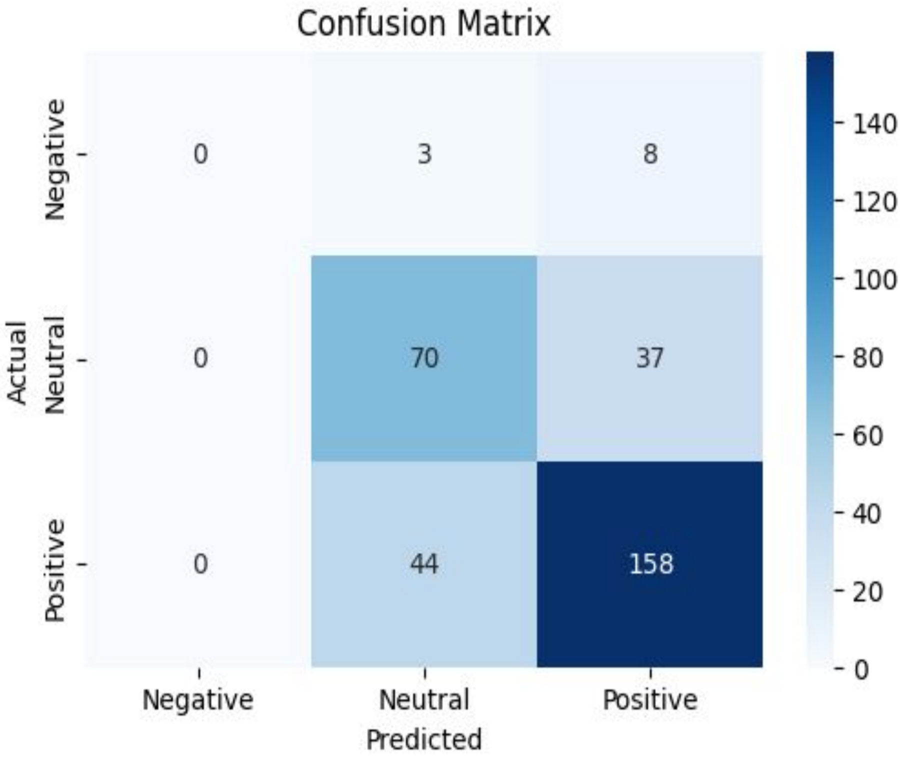
Demo Test

[22]

```python
# Ensure dataset has the necessary columns
if "reviews.text" not in df.columns:
    raise ValueError("Dataset does not contain a 'review' column.")

# Select a random review from the dataset
random_review = df.sample(1)  # Pick a random row
review_text = random_review["reviews.text"].values[0]  # Extract review text

# Display the randomly selected review
print("\nRandomly Selected Review:")
print(review_text)

# Tokenize and encode the selected review for BERT
encoded_input = tokenizer.encode_plus(
    review_text, return_tensors="pt", max_length=128, truncation=True, padding="max_length"
)

# Move tensors to GPU if available
encoded_input = {k: v.cuda() for k, v in encoded_input.items()}

# Get model output
output = model(**encoded_input)

# Determine sentiment label
sentiment_label = torch.argmax(output.logits, dim=1).item()

# Mapping of model output to sentiment labels
sentiment_mapping = {0: "Negative", 1: "Neutral", 2: "Positive"}

# Display predicted sentiment
print(f"\nPredicted Sentiment: {sentiment_mapping[sentiment_label]}")
```

```
Randomly Selected Review:
I love this handheld device especially all the items available for it. Games, books, music, etc. Glad I bought it. Thanks for having it available.

Predicted Sentiment: Positive
```

Heatmap for Basic BERT-Model



Addressing Class Imbalance

Techniques Applied:

- **Class Weighting**: Adjusted loss function to penalize misclassification of minority classes.
- **Random Oversampling**: Increased representation of negative and neutral classes.
- **Balanced CrossEntropyLoss**: Weighted losses based on class frequency.

```
# Extract unique sentiment labels from the dataset
classes = np.unique(df["sentiment"].values)  # Ensure all present labels are included

# Compute class weights
class_weights = compute_class_weight(class_weight="balanced", classes=classes, y=df["sentiment"].values)


# Convert to tensor
class_weights = torch.tensor(class_weights, dtype=torch.float)

boost_factor = 5  # Increase weight impact
class_weights = class_weights * boost_factor

# Apply weighted CrossEntropyLoss
loss_fn = torch.nn.CrossEntropyLoss(weight=class_weights)

# Move to GPU if available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
class_weights = class_weights.to(device)

print("Class Weights:", class_weights)
```

```
Class Weights: tensor([ 5.3663, 21.4651,  2.7243], device='cuda:0')
```

Improved Model Performance (After Class Weight)

Accuracy: 66.8% (Decreased from 71%)

F1-Score: 66% (Improved from 48%)

Confusion Matrix Improvements:

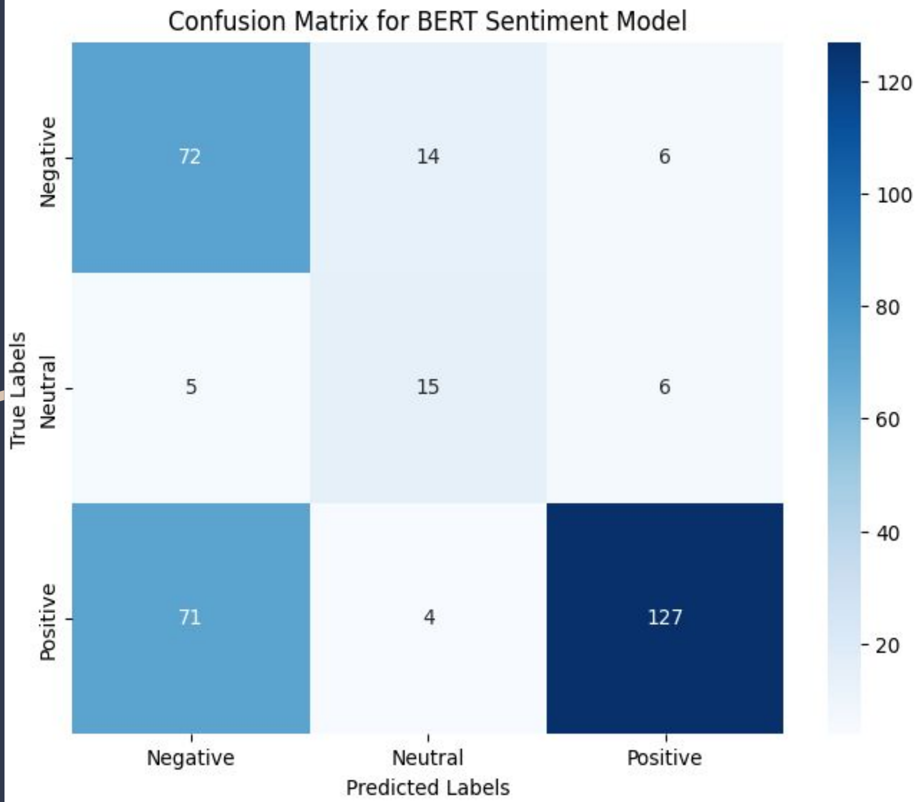
- Better prediction across all classes.
- **Negative and neutral sentiment detection improved.**
- More balanced classification.

Accuracy: 0.66875

Classification Report:

	precision	recall	f1-score	support
0	0.49	0.78	0.60	92
1	0.45	0.58	0.51	26
2	0.91	0.63	0.74	202
accuracy			0.67	320
macro avg	0.62	0.66	0.62	320
weighted avg	0.75	0.67	0.68	320

HeatMap
(After Class Weight)



Sentiment Distribution (Before vs After)

Before Class Weighting:

- Majority of predictions were classified in the positive sentiment.
- The Negative sentiment was under represented.

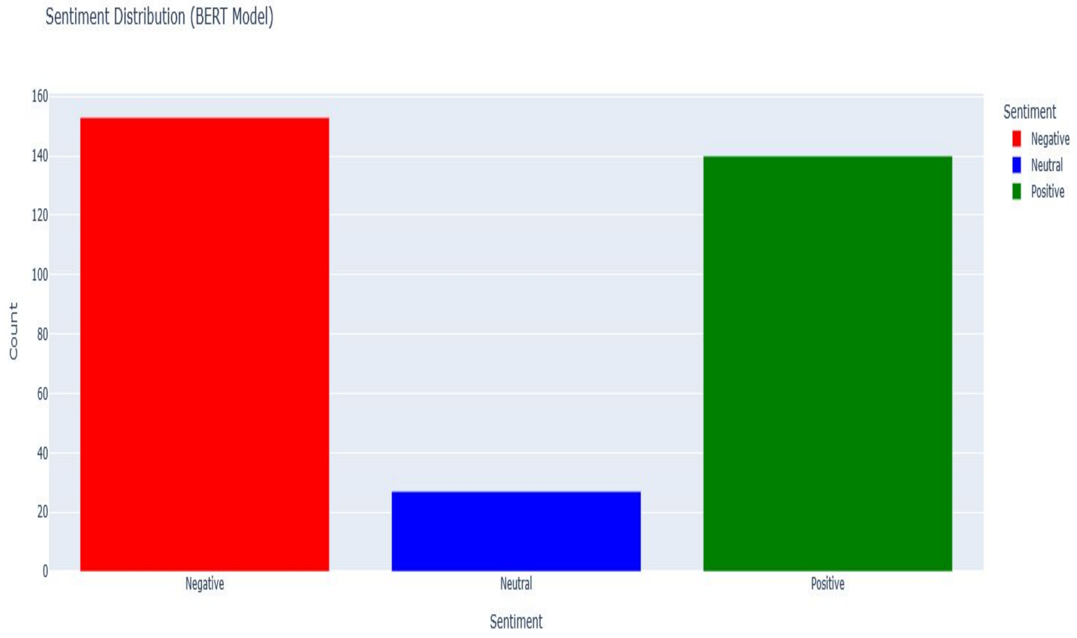
After Class Weighting:

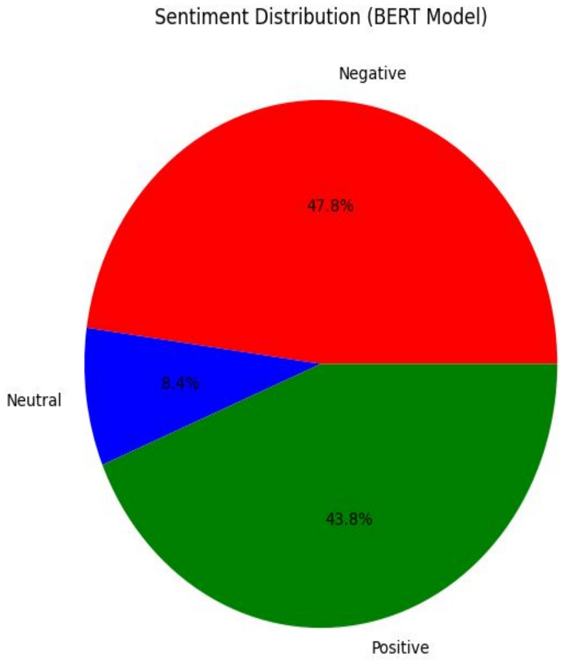
- More balanced distribution across all sentiment labels.

Why does this matter?

- Businesses can capture a broader range of feedback.
- A more even distribution of predictions shows that negative & neutral sentiments are no longer overshadowed by the positive sentiments.

Sentiment Distribution After Class Weights





Model Comparisons

```
SVM Classifier Report:
              precision    recall  f1-score   support

           0       0.50      0.27      0.35        11
           1       0.60      0.69      0.64       107
           2       0.81      0.77      0.79       202

    accuracy                           0.72       320
   macro avg       0.64      0.58      0.60       320
weighted avg       0.73      0.72      0.73       320

Confusion Matrix for SVM:
[[  3   4   4]
 [  1  74  32]
 [  2  45 155]]
```

```
Accuracy: 0.71
Precision: 0.69
Recall: 0.71
F1-Score: 0.70

Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        11
           1       0.60      0.65      0.62       107
           2       0.78      0.78      0.78       202

    accuracy                           0.71       320
   macro avg       0.46      0.48      0.47       320
weighted avg       0.69      0.71      0.70       320
```

```
Random Forest Classifier Report:
              precision    recall  f1-score   support

           0       0.67      0.18      0.29        11
           1       0.64      0.64      0.64       107
           2       0.78      0.81      0.80       202

    accuracy                           0.73       320
   macro avg       0.69      0.54      0.57       320
weighted avg       0.73      0.73      0.72       320

Confusion Matrix for Random Forest:
[[  2   1   8]
 [  1  68  38]
 [  0  38 164]]
```

```
Accuracy: 0.66875

Classification Report:
              precision    recall  f1-score   support

           0       0.49      0.78      0.60        92
           1       0.45      0.58      0.51        26
           2       0.91      0.63      0.74       202

    accuracy                           0.67       320
   macro avg       0.62      0.66      0.62       320
weighted avg       0.75      0.67      0.68       320
```

Conclusion

Conclusion:

- The pre-trained BERT model, is capable of understanding nuanced sentiment, but over-predicts positive sentiment due to their being an imbalance.
- The traditional ML models established a baseline but struggled with class imbalance also.
- Incorporating class weighting and oversampling led to more balanced performance with improved F1-scores for negative and neutral labels.