COURSEWORK 1

MODULE TITLE- INTELLIGENT DATA AND TEXT ANALYTICS

MODULE CODE- M33147

TOPIC- NLP Workflow for Text Data: Preprocessing, Classification, and Topic Modeling

Submitted By-
Student ID: UP2280648

DATE OF SUBMISSION- 20th January, 2025

Table of Contents

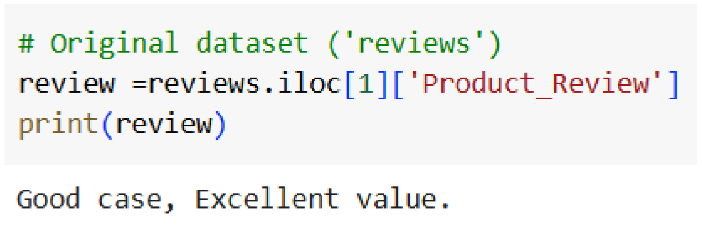
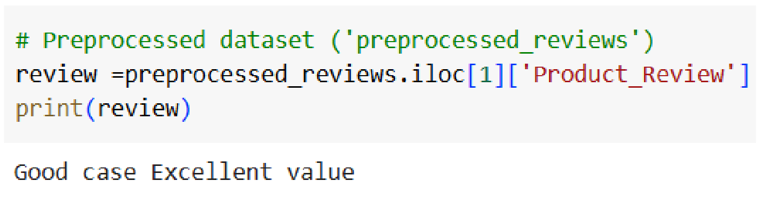
Chapter 1. Text Preprocessing

Firstly, the tab-separated text file **'amazon_cells_labelled.txt'** is read into a pandas DataFrame and stored in the variable named **'reviews'**. The 'reviews' DataFrame has two columns- **'Product_Review'**, which contains customer feedback, and **'Label',** which indicates whether the feedback is negative (0) or positive (1). A copy of the **'reviews'** DataFrame is created and stored in the variable named **'preprocessed_reviews'**. All the subsequent preprocessing methods will be applied on the **'preprocessed_reviews'** DataFrame and the changes to the text will be compared with the original un-processed DataFrame **'reviews'.**

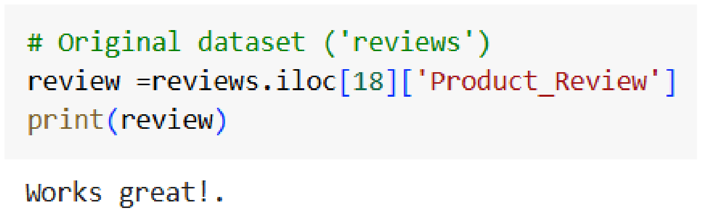
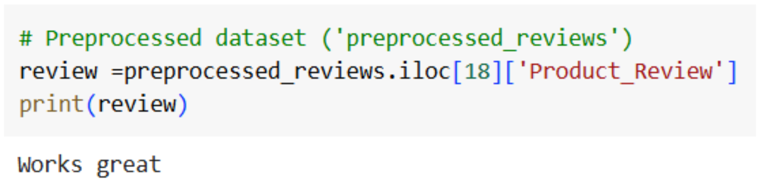
1.1 Removing Punctuations

Table 1: *Results Demonstrating the Removal of Punctuation*

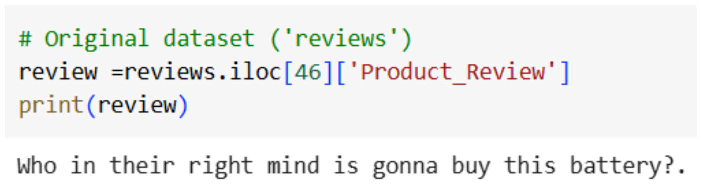
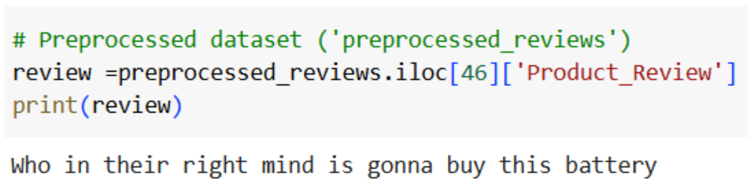
Original Text	Preprocessed Text
Example 1	


`# Original dataset ('reviews')`
`review =reviews.iloc[1]['Product_Review']`
`print(review)`
 	
`Good case, Excellent value.`
	
	`# Preprocessed dataset ('preprocessed_reviews')`
	`review =preprocessed_reviews.iloc[1]['Product_Review']`
	`print(review)`
	
	`Good case Excellent value`

Example 2	


`# Original dataset ('reviews')`
`review =reviews.iloc[18]['Product_Review']`
`print(review)`
 	
`Works great!.`
	
	`# Preprocessed dataset ('preprocessed_reviews')`
	`review =preprocessed_reviews.iloc[18]['Product_Review']`
	`print(review)`
	
	`Works great`

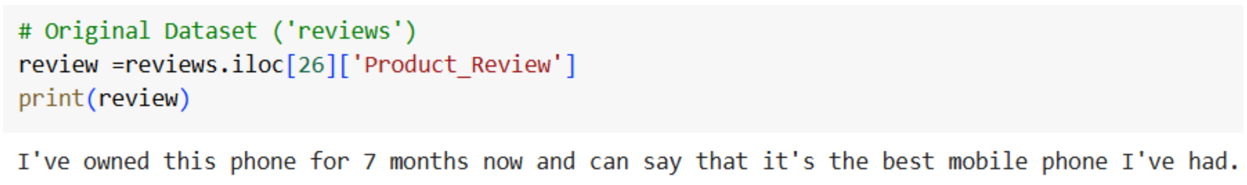
Example 3	

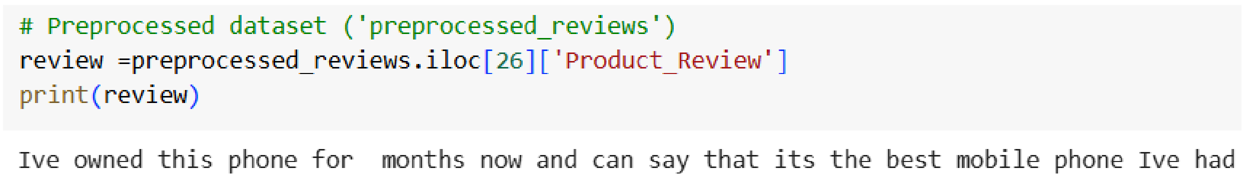

`# Original dataset ('reviews')`
`review =reviews.iloc[46]['Product_Review']`
`print(review)`
 	
`Who in their right mind is gonna buy this battery?.`
	
	`# Preprocessed dataset ('preprocessed_reviews')`
	`review =preprocessed_reviews.iloc[46]['Product_Review']`
	`print(review)`
	
	`Who in their right mind is gonna buy this battery`

1.2 Removing Numbers

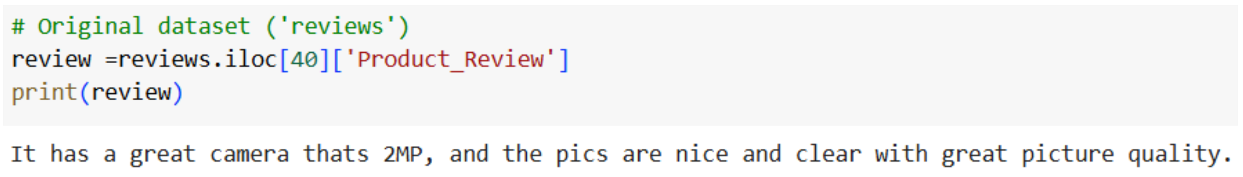
Table 2: *Results Demonstrating the Removal of Numbers*

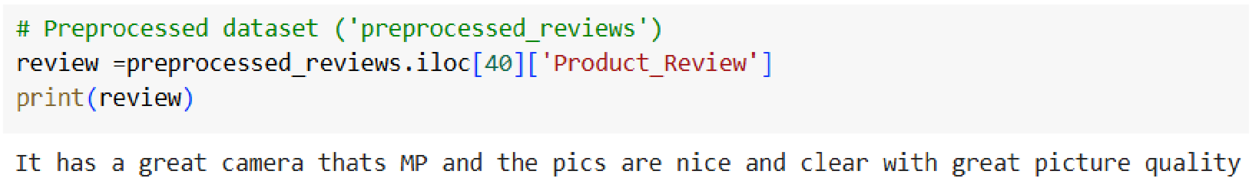
Example 1	

Original Text	 `# Original Dataset ('reviews')`
	`review =reviews.iloc[26]['Product_Review']`
	`print(review)`
	
	`I've owned this phone for 7 months now and can say that it's the best mobile phone I've had.`

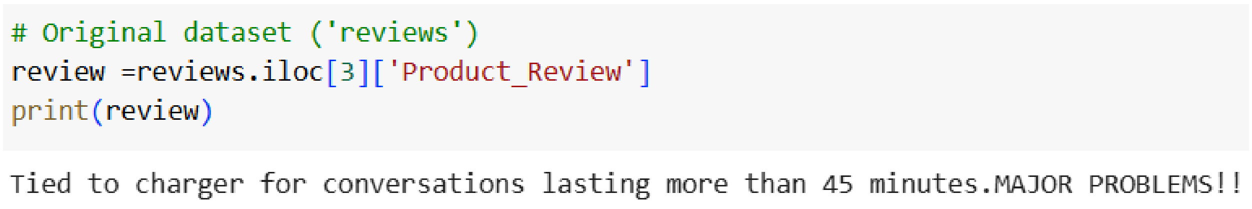
Preprocessed Text	 `# Preprocessed dataset ('preprocessed_reviews')`
	`review =preprocessed_reviews.iloc[26]['Product_Review']`
	`print(review)`
	
	`Ive owned this phone for  months now and can say that its the best mobile phone Ive had`

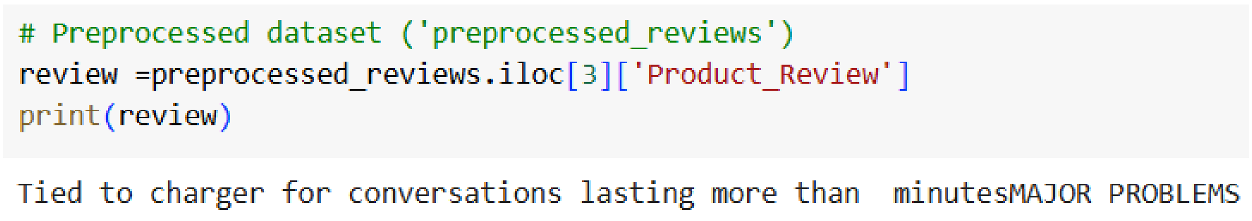
Example 2	

Original Text	 `# Original dataset ('reviews')`
	`review =reviews.iloc[40]['Product_Review']`
	`print(review)`
	
	`It has a great camera thats 2MP, and the pics are nice and clear with great picture quality.`

Preprocessed Text	 `# Preprocessed dataset ('preprocessed_reviews')`
	`review =preprocessed_reviews.iloc[40]['Product_Review']`
	`print(review)`
	
	`It has a great camera thats MP and the pics are nice and clear with great picture quality`

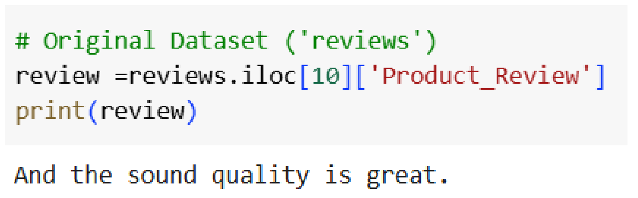
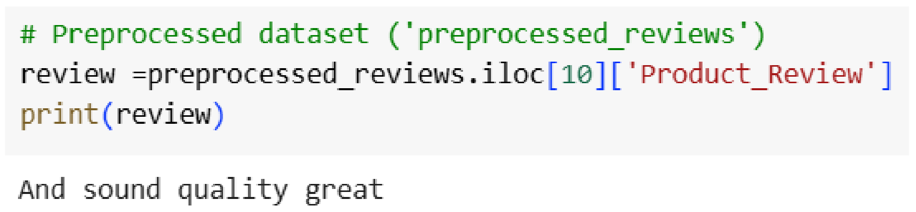
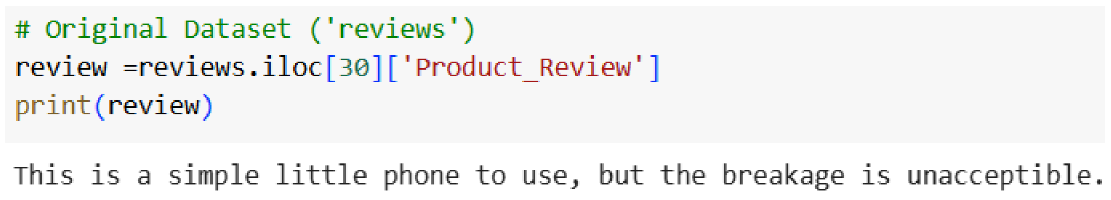
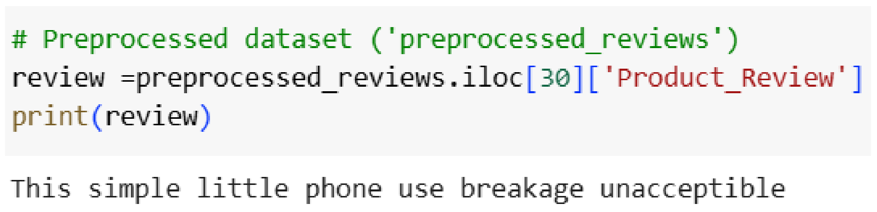
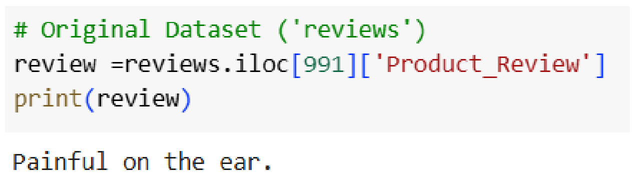
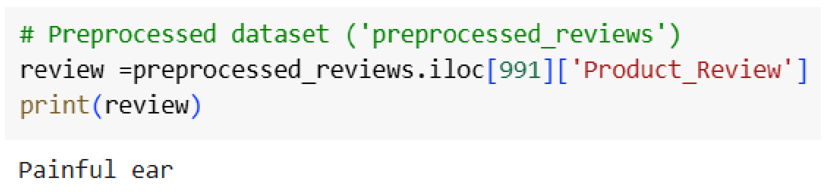
Example 3	

Original Text	 `# Original dataset ('reviews')`
	`review =reviews.iloc[3]['Product_Review']`
	`print(review)`
	
	`Tied to charger for conversations lasting more than 45 minutes.MAJOR PROBLEMS!!`

Preprocessed Text	 `# Preprocessed dataset ('preprocessed_reviews')`
	`review =preprocessed_reviews.iloc[3]['Product_Review']`
	`print(review)`
	
	`Tied to charger for conversations lasting more than  minutesMAJOR PROBLEMS`

1.3 Removing Stop Words

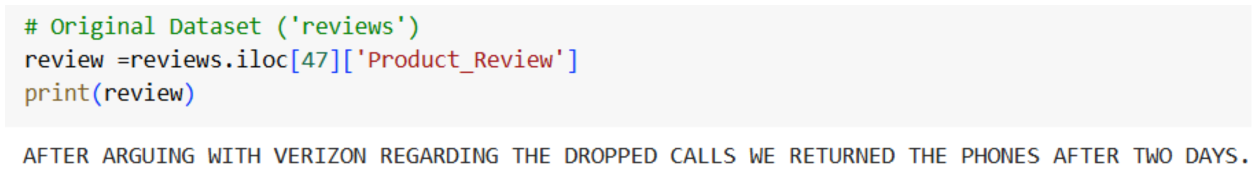
Table 3: *Results Demonstrating the Removal of Stop Words*

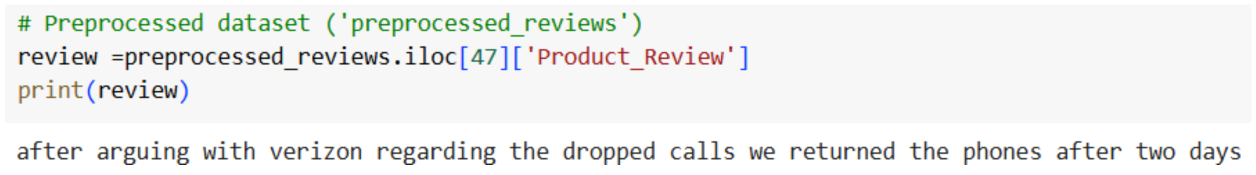
Example 1	
Original Text	 `# Original Dataset ('reviews')` `review =reviews.iloc[10]['Product_Review']` `print(review)` `And the sound quality is great.`
Preprocessed Text	 `# Preprocessed dataset ('preprocessed_reviews')` `review =preprocessed_reviews.iloc[10]['Product_Review']` `print(review)` `And sound quality great`
Example 2	
Original Text	 `# Original Dataset ('reviews')` `review =reviews.iloc[30]['Product_Review']` `print(review)` `This is a simple little phone to use, but the breakage is unacceptible.`
Preprocessed Text	 `# Preprocessed dataset ('preprocessed_reviews')` `review =preprocessed_reviews.iloc[30]['Product_Review']` `print(review)` `This simple little phone use breakage unacceptible`
Example 3	
Original Text	 `# Original Dataset ('reviews')` `review =reviews.iloc[991]['Product_Review']` `print(review)` `Painful on the ear.`
Preprocessed Text	 `# Preprocessed dataset ('preprocessed_reviews')` `review =preprocessed_reviews.iloc[991]['Product_Review']` `print(review)` `Painful ear`

1.4 Conversion of Texts to Lowercase

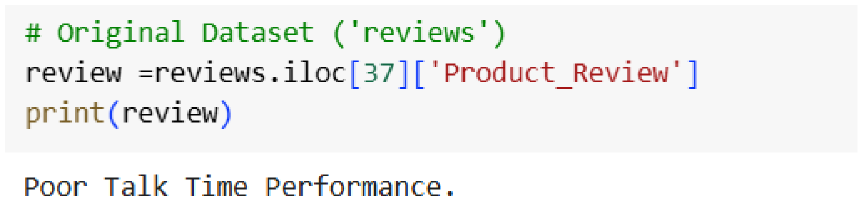
Table 4: *Results Demonstrating the Removal of Stop Words*

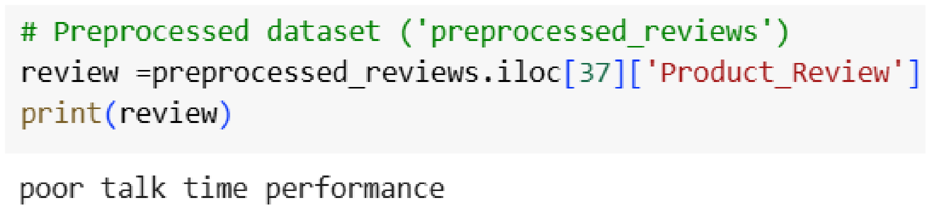
Example 1	

Original Text	 `# Original Dataset ('reviews')`
	`review =reviews.iloc[47]['Product_Review']`
	`print(review)`
	
	`AFTER ARGUING WITH VERIZON REGARDING THE DROPPED CALLS WE RETURNED THE PHONES AFTER TWO DAYS.`

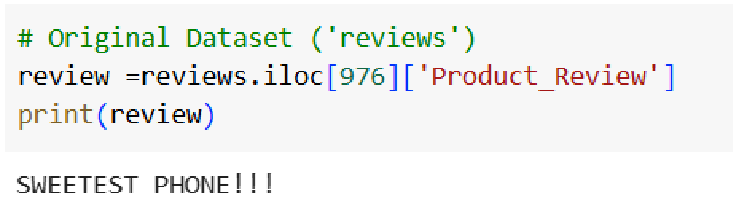
Preprocessed Text	 `# Preprocessed dataset ('preprocessed_reviews')`
	`review =preprocessed_reviews.iloc[47]['Product_Review']`
	`print(review)`
	
	`after arguing with verizon regarding the dropped calls we returned the phones after two days`

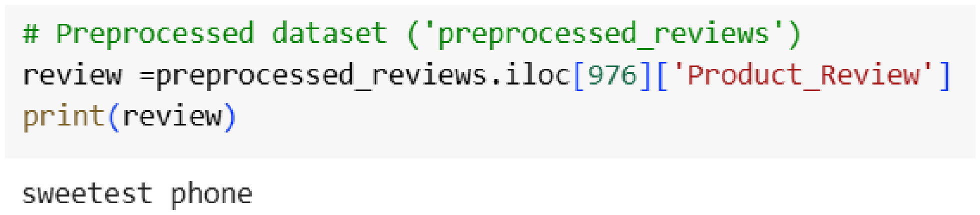
Example 2	

Original Text	 `# Original Dataset ('reviews')`
	`review =reviews.iloc[37]['Product_Review']`
	`print(review)`
	
	`Poor Talk Time Performance.`

Preprocessed Text	 `# Preprocessed dataset ('preprocessed_reviews')`
	`review =preprocessed_reviews.iloc[37]['Product_Review']`
	`print(review)`
	
	`poor talk time performance`

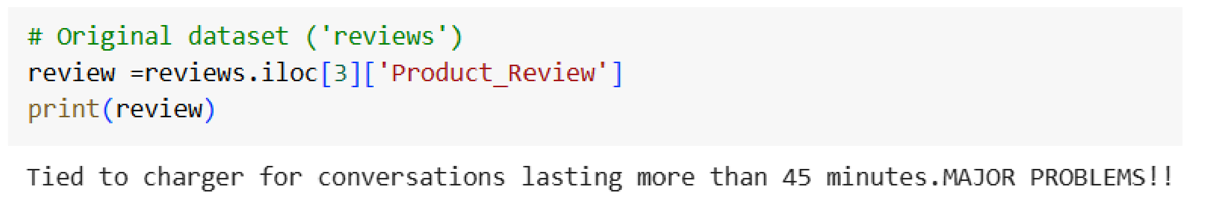
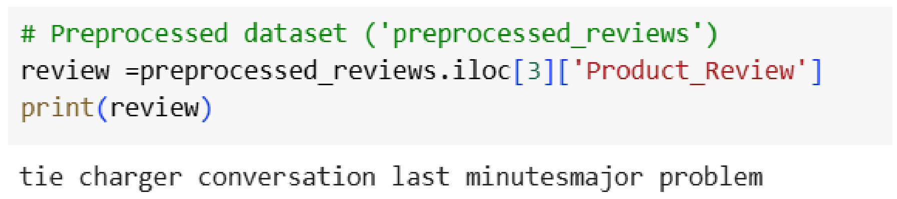
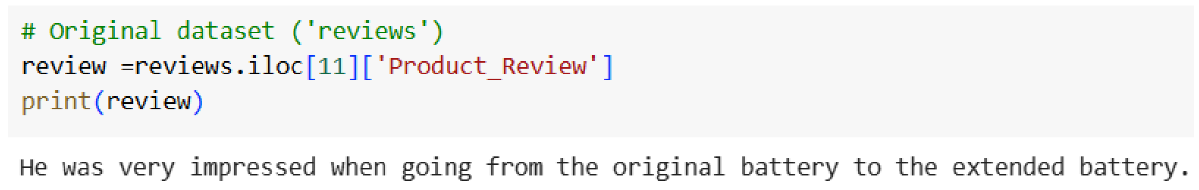
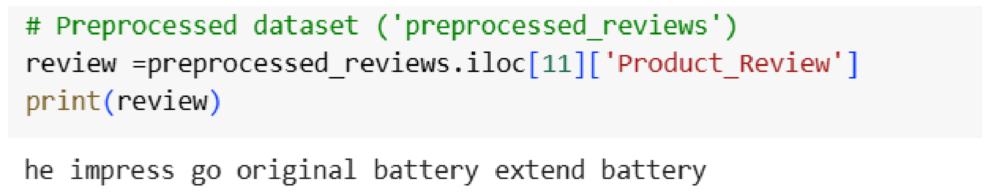
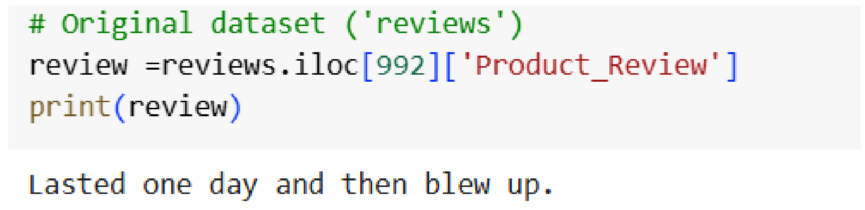
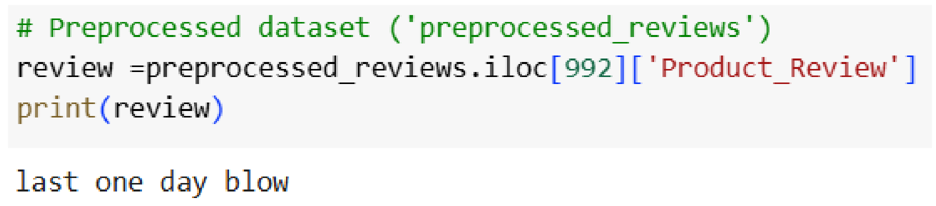
Example 3	

Original Text	 `# Original Dataset ('reviews')`
	`review =reviews.iloc[976]['Product_Review']`
	`print(review)`
	
	`SWEETEST PHONE!!!`

Preprocessed Text	 `# Preprocessed dataset ('preprocessed_reviews')`
	`review =preprocessed_reviews.iloc[976]['Product_Review']`
	`print(review)`
	
	`sweetest phone`

1.5 Lemmatization of Texts

Table 5: *Results Demonstrating the Lemmatization of Texts*

Example 1	
Original Text	 `# Original dataset ('reviews')` `review =reviews.iloc[3]['Product_Review']` `print(review)` `Tied to charger for conversations lasting more than 45 minutes.MAJOR PROBLEMS!!`
Preprocessed Text	 `# Preprocessed dataset ('preprocessed_reviews')` `review =preprocessed_reviews.iloc[3]['Product_Review']` `print(review)` `tie charger conversation last minutesmajor problem`
Example 2	
Original Text	 `# Original dataset ('reviews')` `review =reviews.iloc[11]['Product_Review']` `print(review)` `He was very impressed when going from the original battery to the extended battery.`
Preprocessed Text	 `# Preprocessed dataset ('preprocessed_reviews')` `review =preprocessed_reviews.iloc[11]['Product_Review']` `print(review)` `he impress go original battery extend battery`
Example 3	
Original Text	 `# Original dataset ('reviews')` `review =reviews.iloc[992]['Product_Review']` `print(review)` `Lasted one day and then blew up.`
Preprocessed Text	 `# Preprocessed dataset ('preprocessed_reviews')` `review =preprocessed_reviews.iloc[992]['Product_Review']` `print(review)` `last one day blow`

1.6 Correction of Spellings

Table 6: *Results Demonstrating the Correction of Spellings*

Example 1

Original Text

```
# Original dataset ('reviews')
review =reviews.iloc[30]['Product_Review']
print(review)
```

```
This is a simple little phone to use, but the breakage is unacceptible.
```

Preprocessed Text

```
# Preprocessed dataset ('preprocessed_reviews')
review =preprocessed_reviews.iloc[30]['Product_Review']
print(review)
```

```
this simple little phone use breakage unacceptable
```

Example 2

Original Text

```
# Original dataset ('reviews')
review =reviews.iloc[952]['Product_Review']
print(review)
```

```
Very satisifed with that.
```

Preprocessed Text

```
# Preprocessed dataset ('preprocessed_reviews')
review =preprocessed_reviews.iloc[952]['Product_Review']
print(review)
```

```
very satisfied
```

Example 3

Original Text

```
# Original dataset ('reviews')
review =reviews.iloc[28]['Product_Review']
print(review)
```

```
People couldnt hear me talk and I had to pull out the earphone and talk on the phone.
```

Preprocessed Text

```
# Preprocessed dataset ('preprocessed_reviews')
review =preprocessed_reviews.iloc[28]['Product_Review']
print(review)
```

```
people could not hear talk I pull earphone talk phone
```

Table 7: *Summary and Analysis of the Text Preprocessing Methods*

Preprocessing Method	Description	Example
Removing Punctuations (Refer Table 1) --- **Technique Used:** A 'string.punctuation' method (which includes common punctuation characters) is used with 'str.maketrans' to create a translation table that removes these punctuation characters from the 'Product_Review' column in the 'preprocessed_reviews' DataFrame.	This method involves removing unnecessary punctuation and symbols from the text, such as commas, periods, exclamation marks, question marks, etc. ● In **Example 1**, the comma (,) and the period (.) are removed. ● In **Example 2**, the exclamation mark (!) and the period (.) are removed. ● In **Example 3**, the question mark (?) and the period (.) are removed.	**Example 1:** **Input:** Good case, Excellent value. **Output:** Good case Excellent value --- **Example 2:** **Input:** Works great!. **Output:** Works great --- **Example 3:** **Input:** Who in their right mind is gonna buy this battery?. **Output:** Who in their right mind is gonna buy this battery
Removing Numbers (Refer Table 2) --- **Technique Used:** A 'string.digits' method (which includes digits) is used with 'str.maketrans' to create a translation table that removes these digits from the 'Product_Review' column in the 'preprocessed_reviews' DataFrame.	This method involves removing any digits (numeric data) from the text. ● In **Example 1**, the number '7' is removed. ● In **Example 2,** the number '2' is removed. ● In **Example 3,** the number '45' is removed.	**Example 1:** **Input:** I've owned this phone for 7 months now and can say that it's the best mobile phone I've had. **Output:** Ive owned this phone for months now and can say that its the best mobile phone Ive had --- **Example 2:** **Input:** It has a great camera thats 2MP, and the pics are nice and clear with great picture quality. **Output:** It has a great camera thats MP and the pics are nice and clear with great picture quality --- **Example 3:** **Input:** Tied to charger for conversations lasting more than 45 minutes.MAJOR PROBLEMS!! **Output:** Tied to charger for conversations lasting more than minutesMAJOR PROBLEMS

Removing Stop Words (Refer Table 3) --- **Technique Used:** The NLTK (Natural Language Toolkit) library is used to remove the stop words from the 'Product_Review' column in the 'preprocessed_reviews' DataFrame.	This method filters out common words that carry little meaning eg, 'the', 'is', 'are', 'a', etc. ● In **Example 1,** words such as 'the', and 'is' are removed. ● In **Example 2,** words such as 'is', 'a', 'to', 'but', and 'the' are removed. ● In **Example 3,** words such as 'on', and 'the' are removed.	**Example 1:** **Input:** And the sound quality is great. **Output:** And sound quality great --- **Example 2:** **Input:** This is a simple little phone to use, but the breakage is unacceptible. **Output:** This simple little phone use breakage unacceptible --- **Example 3:** **Input:** Painful on the ear. **Output:** Painful ear
Conversion to Lowercase (Refer Table 4) --- **Technique Used:** The str.lower() method is used to convert all the text in the 'Product_Review' column of the 'preprocessed_reviews' DataFrame to lowercase.	This method converts all the letters to lowercase. ● In **Example 1,** all the letters in the sentence is converted to lowercase. ● In **Example 2,** the first letter of every word was initially in uppercase which is converted into lowercase. ● In **Example 3,** all the letters in the sentence is converted to lowercase.	**Example 1** **Input:** AFTER ARGUING WITH VERIZON REGARDING THE DROPPED CALLS WE RETURNED THE PHONES AFTER TWO DAYS. **Output:** after arguing with verizon regarding the dropped calls we returned the phones after two days --- **Example 2:** **Input:** Poor Talk Time Performance. **Output:** poor talk time performance --- **Example 3:** **Input:** SWEETEST PHONE!!! **Output:** sweetest phone

Lemmatization (Refer Table 5) ——— **Technique Used:** The 'spaCy' library is used to lemmatize the text in the 'Product_Review' column of the 'preprocessed_reviews' DataFrame. Each text is processed using the 'en_core_web_sm' language model of the spaCy's library. The 'lemma_' attribute is used to transform each word with its base or root form.	This method transforms the words to their base or root form while preserving their meanings. ● In **Example 1**, words like "tied" and "lasting" are reduced to "tie" and "last". ● In **Example 2**, words like "impressed" and "going" are reduced to "impress" and "go". ● In **Example 3**, words like "lasted" and "blew" are reduced to "last" and "blow".	**Example 1:** **Input:** Tied to charger for conversations lasting more than 45 minutes.MAJOR PROBLEMS!! **Output:** tie charger conversation last minutesmajor problem ——— **Example 2:** **Input:** He was very impressed when going from the original battery to the extended battery. **Output:** he impress go original battery extend battery ——— **Example 3:** **Input:** Lasted one day and then blew up. **Output:** last one day blow
Correction of Spellings (Refer Table 6) ——— The 'SpellChecker' library is used to correct misspelled words in the 'Product_Review' column of the 'preprocessed_reviews' DataFrame. The 'correction' method replaces any misspelled word with its correct spelling.	This method corrects misspelled words to their correct spelling. ● In **Example 1**, the word "unacceptible" is corrected to "unacceptable". ● In **Example 2**, the word "satisifed" is corrected to "satisfied". ● In **Example 3**, the word "couldnt" is corrected to "could not".	**Example 1:** **Input:** This is a simple little phone to use, but the breakage is unacceptible. **Output:** this simple little phone use breakage unacceptable ——— **Example 2:** **Input:** Very satisifed with that. **Output:** very satisfied ——— **Example 3:** **Input:** People couldnt hear me talk and I had to pull out the earphone and talk on the phone. **Output:** people could not hear talk I pull earphone talk phone

Chapter 2. Text Classification with Bag-of-Words Across Multiple Algorithms

2.1 Data Preprocessing for Classification

Table 8: *Data Preprocessing Techniques*

Preprocessing Technique	Description
Bag-of-Words Representation of the Textual Data	The CountVectorizer method from the scikit learn library is used to convert the text present in the 'Product_Review' column of the 'preprocessed_reviews' DataFrame into bag-of-words representation. The resulting DataFrame is stored in the variable 'df' where each column is a unique word feature, with an additional 'Label' column indicating the corresponding sentiment.
Segregating 'Features' and 'Target' Column	The 'Label' column in the 'df' DataFrame is the target column. The features include all the columns of the DataFrame except the 'Label' column. The features are stored in the variable named 'X' and the target column is stored in the variable named 'y'.
Splitting the data into 'Train' and 'Test'	The features 'X' and target column (y) are split into 75% for training and 25% for testing where 'X_train' contains 75% of the features for training, 'X_test' contains 25% of the features for testing, 'y_train' contains 75% of the target data for training and 'y_test' contains 25% of the target data for testing.

2.2 Classification Models

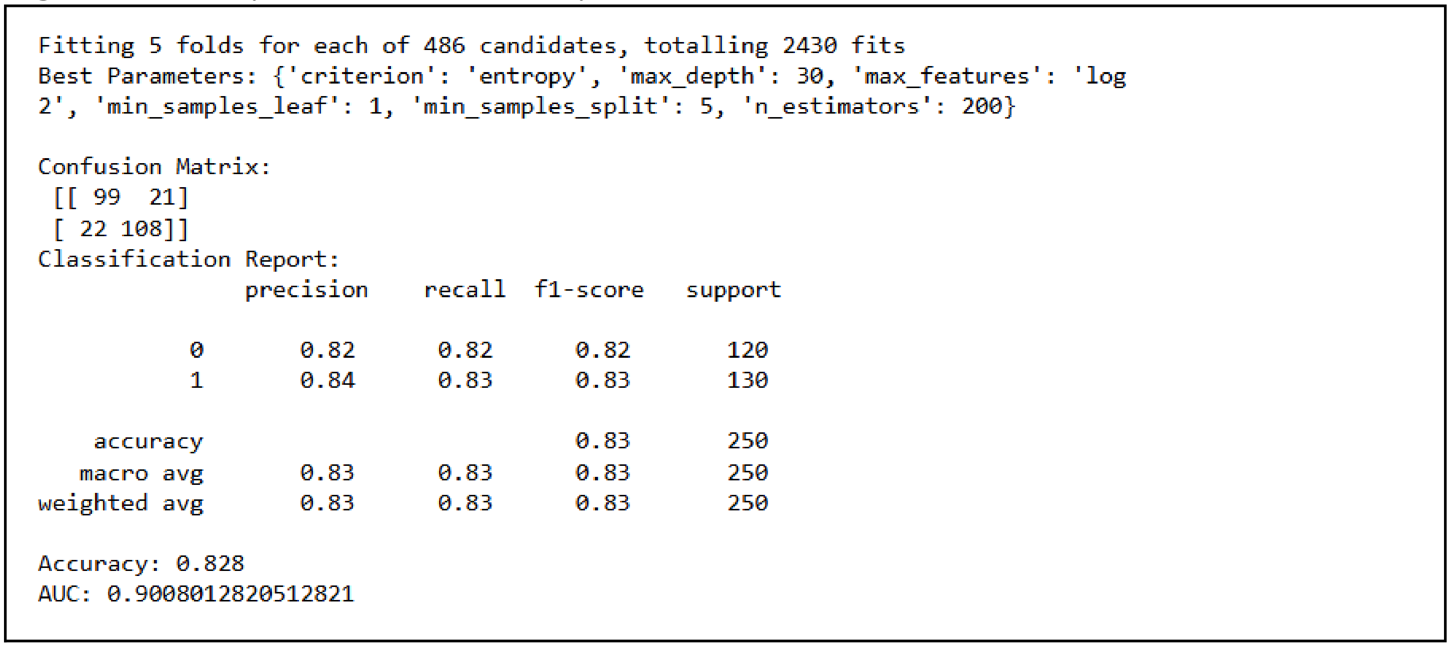
2.2.1 Random Forest Classifier: Hyperparameter Tuning and Model Evaluation

- The RandomForestClassifier and GridSearchCV are imported from the sklearn library.
- The hyperparameters chosen for GridSearchCV and the corresponding values are stored in a variable named 'parameters' which are as follows-
  - n_estimators: [100, 200, 300].
  - max_depth: [10, 20, 30].
  - min_samples_split: [2, 5, 10].

    - min_samples_leaf: [1, 2, 4].
    - criterion: ['gini', 'entropy', 'log_loss'].
    - max_features: ['sqrt', 'log2']
- The RandomForestClassifier is initialized as RF with random_state as 0 for the reproducibility of the code.
- A GridSearchCV object is created with the Random Forest model (RF), parameters, and a 5-fold cross-validation (cv=5). The scoring metric used for evaluation is accuracy.
- The grid search is performed on training data i.e X_train and y_train to find the optimal combination of hyperparameters
- The best model is then identified using grid_search.best_estimator_, and predictions are made using the test data (X_test) based on the best parameters. The predictions made are stored as y_pred.
- The performance of the model is evaluated using a confusion matrix, classification report, accuracy score with the help of y_test (actual values) and y_pred (predicted values) and Receiver Operating Characteristics curve.

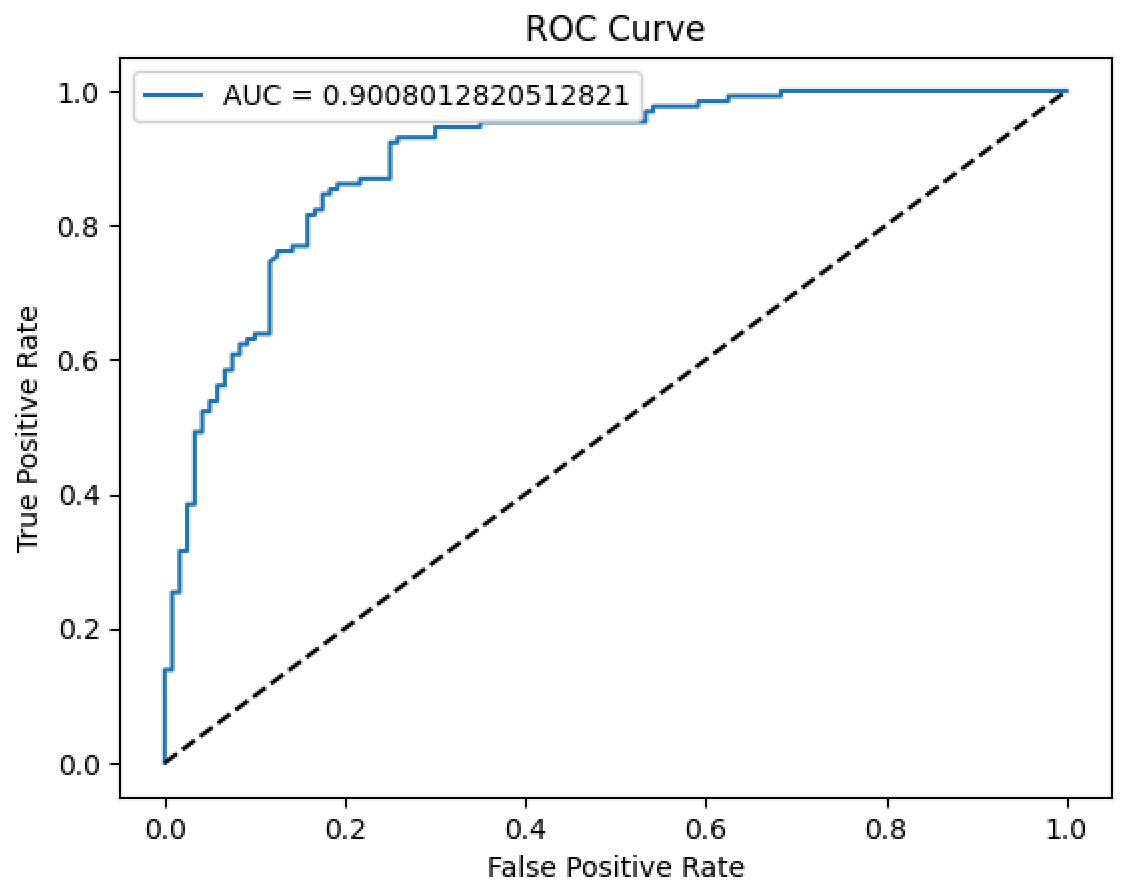
Result:

Figure 1: *Result of Random Forest Classifier*



```
Fitting 5 folds for each of 486 candidates, totalling 2430 fits
Best Parameters: {'criterion': 'entropy', 'max_depth': 30, 'max_features': 'log
2', 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 200}

Confusion Matrix:
 [[ 99  21]
 [ 22 108]]
Classification Report:
               precision    recall  f1-score   support

           0       0.82      0.82      0.82       120
           1       0.84      0.83      0.83       130

    accuracy                           0.83       250
   macro avg       0.83      0.83      0.83       250
weighted avg       0.83      0.83      0.83       250

Accuracy: 0.828
AUC: 0.9008012820512821
```

Figure 2: *Receiver Operating Charateristics*



Analysis of the Result

1. **Best Hyperparameters**

Table 9: *Table Showcasing Best Hyperparameters*

Hyperparameter	Description	Best Value
criterion	Function which is used to measure the quality of a split	entropy
max_depth	It is the maximum depth of each tree	30
max_features	Number of features for best split	log2
min_samples_leaf	Minimum number of samples required to be at the leaf node	1

min_samples_split	Minimum number of samples required to split an internal node	5
n_estimators	Total number of trees in the forest	200

2. **Analysis of the Confusion Matrix:**

Table 10: *Table showing the interpretation of the confusion matrix*

Actual/ Predicted	**Predicted Label 0 (Negative Sentiment)**	**Predicted Label 1 (Positive Sentiment)**
Actual Label 0 (Negative Sentiment)	99 instances of label 0 are correctly classified (True Negatives)	21 instances of label 0 are misclassified as label 1 (False Positives)
Actual Label 1 (Positive Sentiment)	22 instances of label 1 are misclassified as label 0 (False Negatives)	108 instances of label 1 are correctly classified (True Positives)

3. **Analysis of Precision, Recall and F1-Score**

Table 11: *Table showing interpretation of the precision, recall and F1-Score of the two classes*

Target Labels (Label)	**Analysis**
0 (Negative Sentiment)	A precision of 0.82 suggests that when the model classifies label 0, it is correct 82% of the time. A recall of 0.82 suggests that the model is able to capture 82% of actual label 0 instances. An f-1 score of 0.82 suggests the overall balance between precision and recall. There are 120 actual instances of label 0 in total.
1 (Positive Sentiment)	A precision of 0.84 suggests that when the model classifies label 1, it is correct 84% of the time. A recall of 0.83 suggests that the model is able to capture 83% of actual label 1 instances. An f-1 score of 0.83 suggests the overall balance between precision and recall. There are 130 actual instances of label 1 in total.

4. **Analysis of Accuracy, Macro Average, and Weighted Average**

Table 12: *Table showing interpretation of Accuracy, Macro Average and Weighted Average of the trained model*

Accuracy	An accuracy of **82.8%** suggests that the model correctly classifies **82.8%** of the total instances.
Macro Average	This is the average of precision, recall, and F1-score across all classes, it is calculated without considering the support of each class. The macro average of **83%**, for precision, recall and f1 score suggests that on average, both the classes perform at a similar level.
Weighted Average	This is the average of precision, recall, and F1-score and it takes into account the support of each class. The weighted average in this case is **83%**, reflecting the overall performance weighted by class size.

5. **Analysis of the ROC Curve (Receiver Operating Characteristic Curve)**

Table 13: *Table showing interpretation of the ROC Curve*

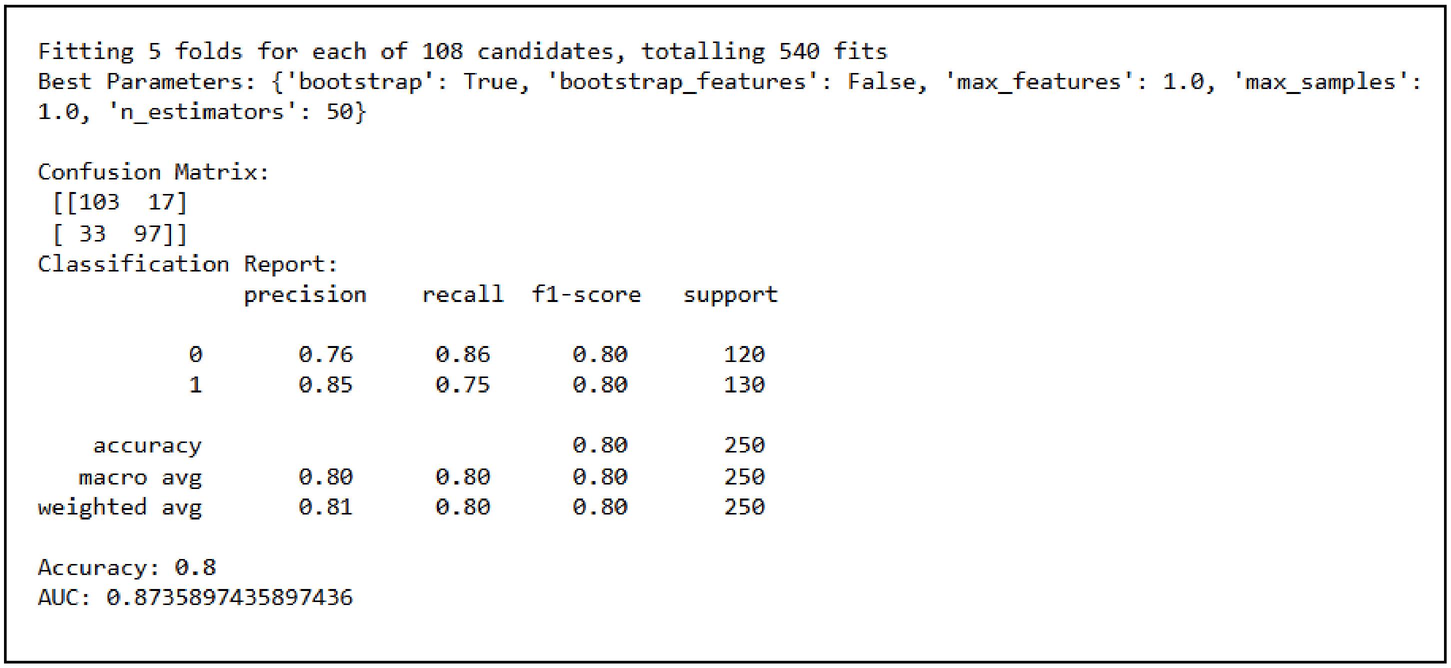
Aspect	**Observation**
AUC (Area Under the Curve)	The area under the ROC curve is 0.900 (AUC score). This score indicates that the model can effectively distinguish between the positive and negative class in our dataset.
Low False Positive Rate	The ROC curve remains close to the top-left which indicates minimum false positives
High True Positive Rate	The model effectively captures a large number of true positive instances.
Above the Diagonal	The ROC curve lies significantly above the diagonal line which means that the model is not randomly guessing the outcomes.

2.2.2 Bagging Classifier: Hyperparameter Tuning and Model Evaluation

- The BaggingClassifier is imported from the sklearn library.
- The hyperparameters chosen for GridSearchCV and the corresponding values are stored in a variable named ‘parameters’ which are as follows-
  - n_estimators: [50, 100, 150]
  - max_samples: [0.5, 0.7, 1.0]
  - max_features: [0.5, 0.7, 1.0]
  - bootstrap: [True, False]
  - bootstrap_features: [True, False]
- The BaggingClassifier is initialized as BC with random_state as 0 for the reproducibility of the results.
- A GridSearchCV object is created with the BaggingClassifier (BC), the above-mentioned parameters, and a 5-fold cross-validation (cv=5). The scoring metric used for evaluation is accuracy.
- The grid search is performed on training data i.e X_train and y_train to find the optimal combination of hyperparameters
- The best model is then identified using grid_search.best_estimator_, and predictions are made using the test data (X_test) based on the best parameters. The predictions made are stored as y_pred.
- The performance of the model is evaluated using a confusion matrix, classification report, accuracy score with the help of y_test (actual values) and y_pred (predicted values) and Receiver Operating Characteristics curve.

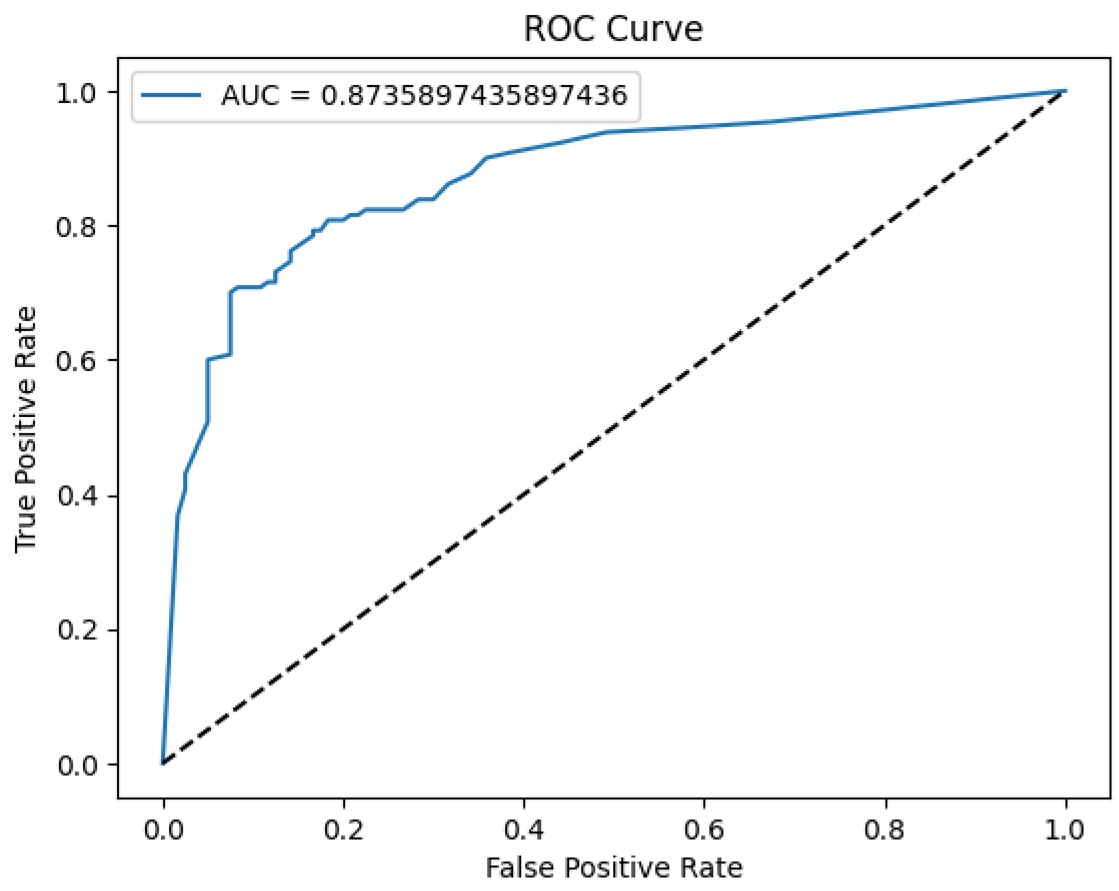
Result:

Figure 3: *Result of Bagging Classifier*



```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
Best Parameters: {'bootstrap': True, 'bootstrap_features': False, 'max_features': 1.0, 'max_samples':
1.0, 'n_estimators': 50}

Confusion Matrix:
 [[103  17]
 [ 33  97]]
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.86      0.80       120
           1       0.85      0.75      0.80       130

    accuracy                           0.80       250
   macro avg       0.80      0.80      0.80       250
weighted avg       0.81      0.80      0.80       250

Accuracy: 0.8
AUC: 0.8735897435897436
```

Figure 4: ***Receiver Operating Charateristics***



Analysis of the Result:

1. **Best Hyperparameters**

Table 14: *Table Showcasing Best Hyperparameters*

Hyperparameter	**Description**	**Best Value**
bootstrap	Whether to sample data with replacement for training	True
bootstrap_features	Whether to sample features with replacement for training	False
max_features	Proportion of features sampled for training each base estimator (Decision Tree)	1.0
max_samples	Proportion of dataset sampled for training each base estimator (Decision Tree)	1.0

n_estimators	Total number of trees in the ensemble model	50

2. Analysis of the Confusion Matrix

Table 15: *Table showing the interpretation of the confusion matrix*

Actual/ Predicted	**Predicted Label 0 (Negative Sentiment)**	**Predicted Label 1 (Positive Sentiment)**
Actual Label 0 (Negative Sentiment)	103 instances of label 0 are correctly classified (True Negatives)	17 instances of label 0 are misclassified as label 1 (False Positives)
Actual Label 1 (Positive Sentiment)	33 instances of label 1 are misclassified as label 0 (False Negatives)	97 instances of label 1 are correctly classified (True Positives)

3. Analysis of Precision, Recall and F1-Score

Table 16: *Table showing interpretation of the precision, recall and F1-Score of the two classes*

Target Labels (Label)	**Analysis**
0 (Negative Sentiment)	A precision of 0.76 suggests that when the model classifies label 0, it is correct 76% of the time. A recall of 0.86 suggests that the model is able to capture 86% of actual label 0 instances. An f-1 score of 0.80 suggests the overall balance between precision and recall. There are 120 actual instances of label 0 in total.
1 (Positive Sentiment)	A precision of 0.85 suggests that when the model classifies label 1, it is correct 85% of the time. A recall of 0.75 suggests that the model is able to capture 75% of actual label 1 instances. An f-1 score of 0.80 suggests the overall balance between precision and recall. There are 130 actual instances of label 1 in total.

4. **Analysis of Accuracy, Macro Average, and Weighted Average**

Table 17: *Table showing interpretation of Accuracy, Macro Average and Weighted Average of the trained model*

Accuracy	An accuracy of **80%** suggests that the model correctly classifies **80%** of the total instances.
Macro Average	This is the average of precision, recall, and F1-score across all classes, it is calculated without considering the support of each class. The macro average of **80%**, for precision, recall and f1 score suggests that on average, both the classes perform at a similar level.
Weighted Average	This is the average of precision, recall, and F1-score and it takes into account the support of each class. The weighted average in this case is **81%** for precision and **80 %** for recall and f1-score, reflecting the overall performance weighted by class size.

5. **Analysis of the ROC Curve (Receiver Operating Characteristic Curve)**

Table 18: *Table showing interpretation of the ROC Curve*

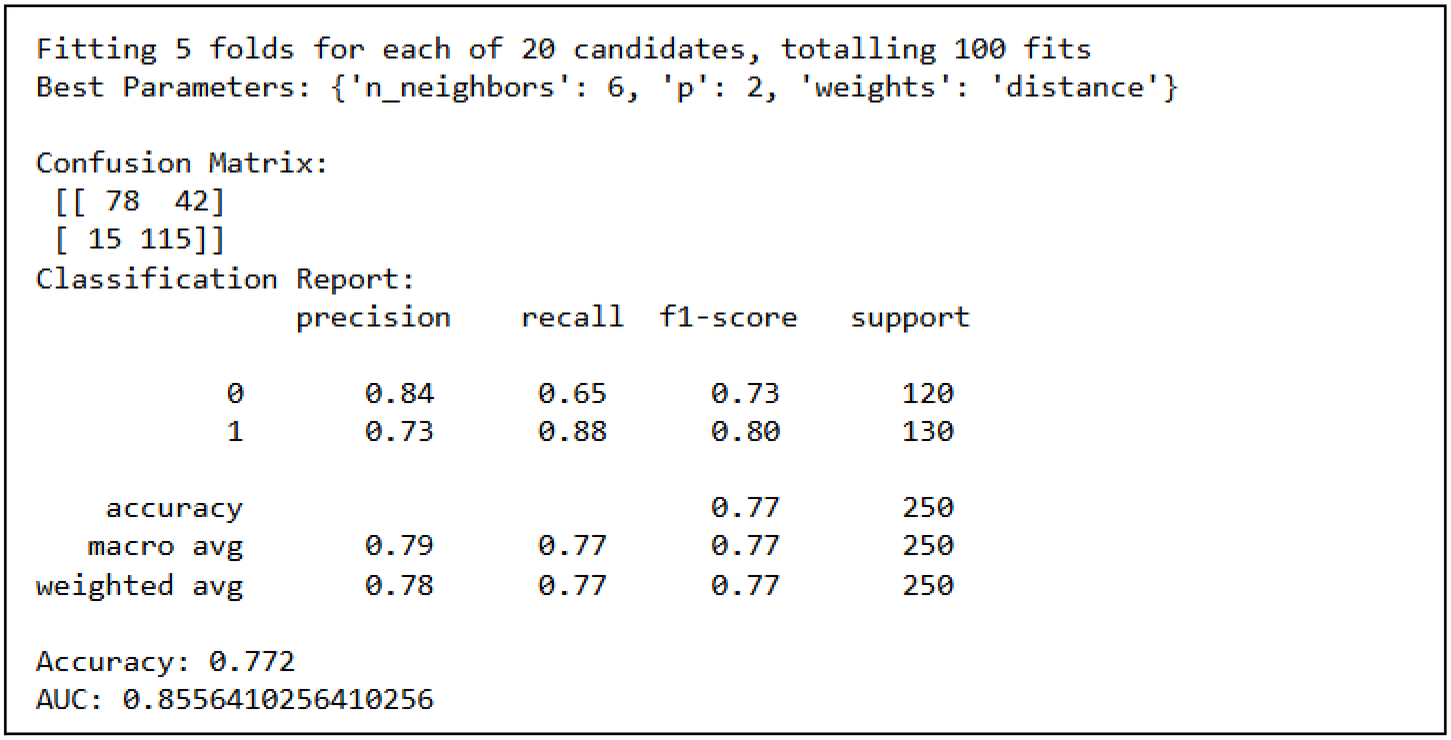
Aspect	**Observation**
AUC (Area Under the Curve)	The area under the ROC curve is 0.873 (AUC score). This score indicates that the model can effectively distinguish between the positive and negative class in our dataset.
Low False Positive Rate	The ROC curve remains close to the top-left which indicates minimum false positives
High True Positive Rate	The model effectively captures large number of true positive instances.
Above the Diagonal	The ROC curve lies significantly above the diagonal line which means that the model is not randomly guessing the outcomes.

2.2.3 KNeighborsClassifier: Hyperparameter Tuning and Model Evaluation

- The KNeighborsClassifier is imported from the sklearn library.
- The hyperparameters chosen for GridSearchCV and the corresponding values are stored in a variable named 'parameters' which are as follows-
    - n_neighbors: [3,4,5,6,7]
    - weights: ['uniform', 'distance']
    - p: [1,2]
- The KNeighborsClassifier is initialized as KNN.
- A GridSearchCV object is created with the KNeighborsClassifier (KNN), the above-mentioned parameters, and a 5-fold cross-validation (cv=5). The scoring metric used for evaluation is accuracy.
- The grid search is performed on training data i.e X_train and y_train to find the optimal combination of hyperparameters
- The best model is then identified using grid_search.best_estimator_, and predictions are made using the test data (X_test) based on the best parameters. The predictions made are stored as y_pred.
- The performance of the model is evaluated using a confusion matrix, classification report, accuracy score with the help of y_test (actual values) and y_pred (predicted values) and Receiver Operating Characteristics curve.

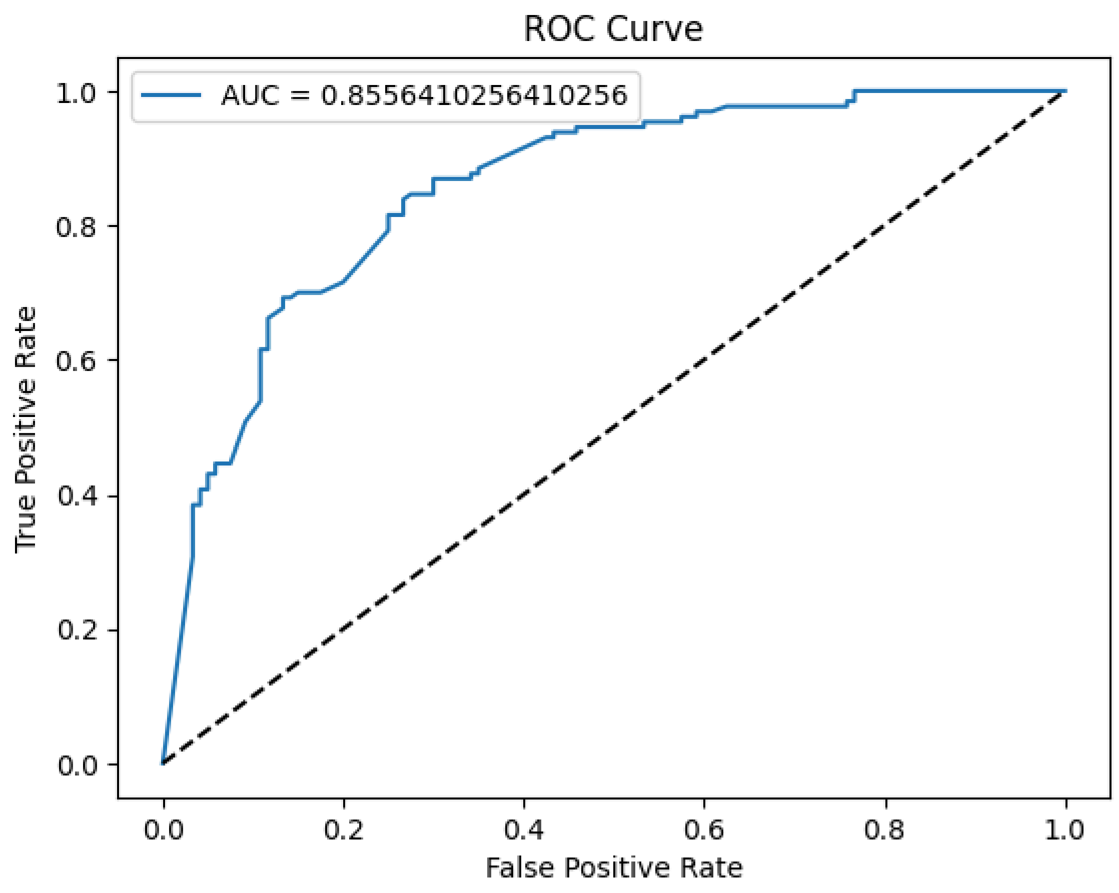
Result:

Figure 5: *Result of KNN Classifier*



```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
Best Parameters: {'n_neighbors': 6, 'p': 2, 'weights': 'distance'}

Confusion Matrix:
 [[ 78  42]
 [ 15 115]]
Classification Report:
               precision    recall  f1-score   support

           0       0.84      0.65      0.73       120
           1       0.73      0.88      0.80       130

    accuracy                           0.77       250
   macro avg       0.79      0.77      0.77       250
weighted avg       0.78      0.77      0.77       250

Accuracy: 0.772
AUC: 0.8556410256410256
```

Figure 6: *Receiver Operating Characteristics*



Analysis of the Result:

1. **Best Hyperparameters**

Table 19: *Table Showcasing Best Hyperparameters*

Hyperparameter	**Description**	**Best Value**
n_neighbors	Number of nearest neighbors	6
p	Distance metric used: 1 for Manhattan distance or 2 for Euclidean distance	2
weights	It is the criteria to weight the neighbor's votes: **uniform** (equal votes) or **distance** (closer ones count more)	distance

2. **Analysis of the Confusion Matrix**

Table 20: *Table showing the interpretation of the confusion matrix*

Actual/ Predicted	**Predicted Label 0 (Negative Sentiment)**	**Predicted Label 1 (Positive Sentiment)**
Actual Label 0 (Negative Sentiment)	78 instances of label 0 are correctly classified (True Negatives)	42 instances of label 0 are misclassified as label 1 (False Positives)
Actual Label 1 (Positive Sentiment)	15 instances of label 1 are misclassified as label 0 (False Negatives)	115 instances of label 1 are correctly classified (True Positives)

3. **Analysis of Precison, Recall and, F1-Score**

Table 21: *Table showing interpretation of the precision, recall and F1-Score of the two classes*

Target Labels (Label)	**Analysis**
0 (Negative Sentiment)	A precision of 0.84 suggests that when the model classifies label 0, it is correct 84% of the time. A recall of 0.65 suggests that the model is able to capture 65%% of actual label 0 instances. An f-1 score of 0.73 suggests the overall balance between precision and recall. There are 120 actual instances of label 0 in total.
1 (Positive Sentiment)	A precision of 0.73 suggests that when the model classifies label 1, it is correct 73% of the time. A recall of 0.88 suggests that the model is able to capture 88% of actual label 1 instances. An f-1 score of 0.80 suggests the overall balance between precision and recall. There are 130 actual instances of label 1 in total.

4. **Analysis of Accuracy, Macro Average, and Weighted Average**

Table 22: *Table showing interpretation of Accuracy, Macro Average and Weighted Average of the trained model*

Accuracy	An accuracy of **77.2%** suggests that the model correctly classifies **77.2%** of the total instances.
Macro Average	This is the average of precision, recall, and F1-score across all classes, it is calculated without considering the support of each class. The macro average of **79%** for precision and

	77% for recall and f1 score suggests that on average, both the classes perform at a similar level.
Weighted Average	This is the average of precision, recall, and F1-score and it takes into account the support of each class. The weighted average in this case is **78%** for precision and **77%** for recall and f1-score reflecting the overall performance weighted by class size.

5. **Analysis of the ROC Curve (Receiver Operating Characteristics)**

Table 23: *Table showing interpretation of the ROC Curve*

Aspect	**Observation**
AUC (Area Under the Curve)	The area under the ROC curve is 0.855 (AUC score). This high score indicates that the model can effectively distinguish between the positive and negative class in our dataset.
Low False Positive Rate	The ROC curve remains close to the top-left which indicates minimum false positives
High True Positive Rate	The model effectively captures large number of true positive instances.
Above the Diagonal	The ROC curve lies significantly above the diagonal line which means that the model is not randomly guessing the outcomes.

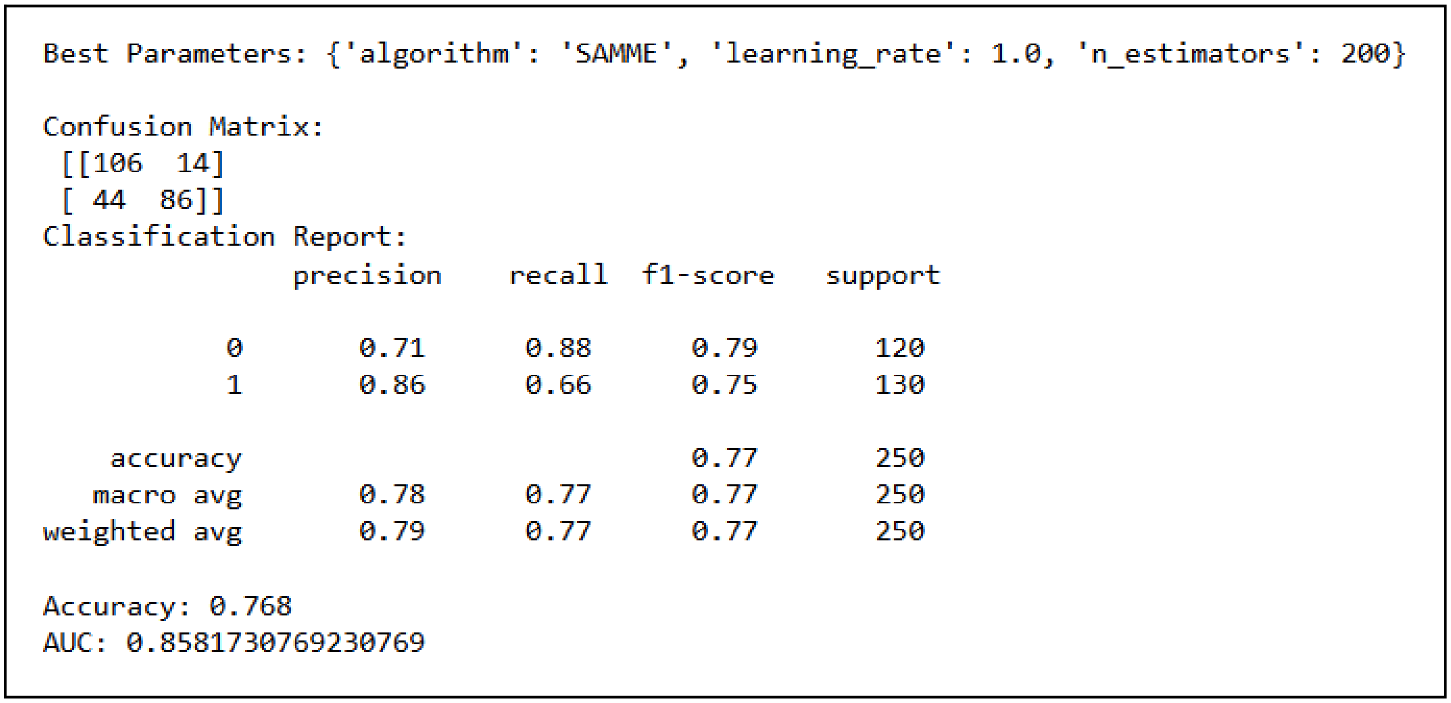
2.2.4 AdaBoost Classifier

- The AdaBoostClassifier is imported from the sklearn library.

- The hyperparameters chosen for GridSearchCV and the corresponding values are stored in a variable named ‘parameters’ which are as follows-
    - n_estimators: [100, 150, 200, 250]
    - learning_rate: [0.001, 0.01, 0.1, 1.0, 2.0]
    - algorithm: [‘SAMME’, ‘SAMME.R’]
- The AdaBoostClassifier is initialized as ABC with random_state as 0 for the reproducibility of the results
- A GridSearchCV object is created with the AdaBoostClassifier (ABC), the above-mentioned parameters, and a 5-fold cross-validation (cv=5). The scoring metric used for evaluation is accuracy.
- The grid search is performed on training data i.e X_train and y_train to find the optimal combination of hyperparameters
- The best model is then identified using grid_search.best_estimator_, and predictions are made using the test data (X_test) based on the best parameters. The predictions made are stored as y_pred.
- The performance of the model is evaluated using a confusion matrix, classification report, accuracy score with the help of y_test (actual values) and y_pred (predicted values) and Receiver Operating Characteristics curve.

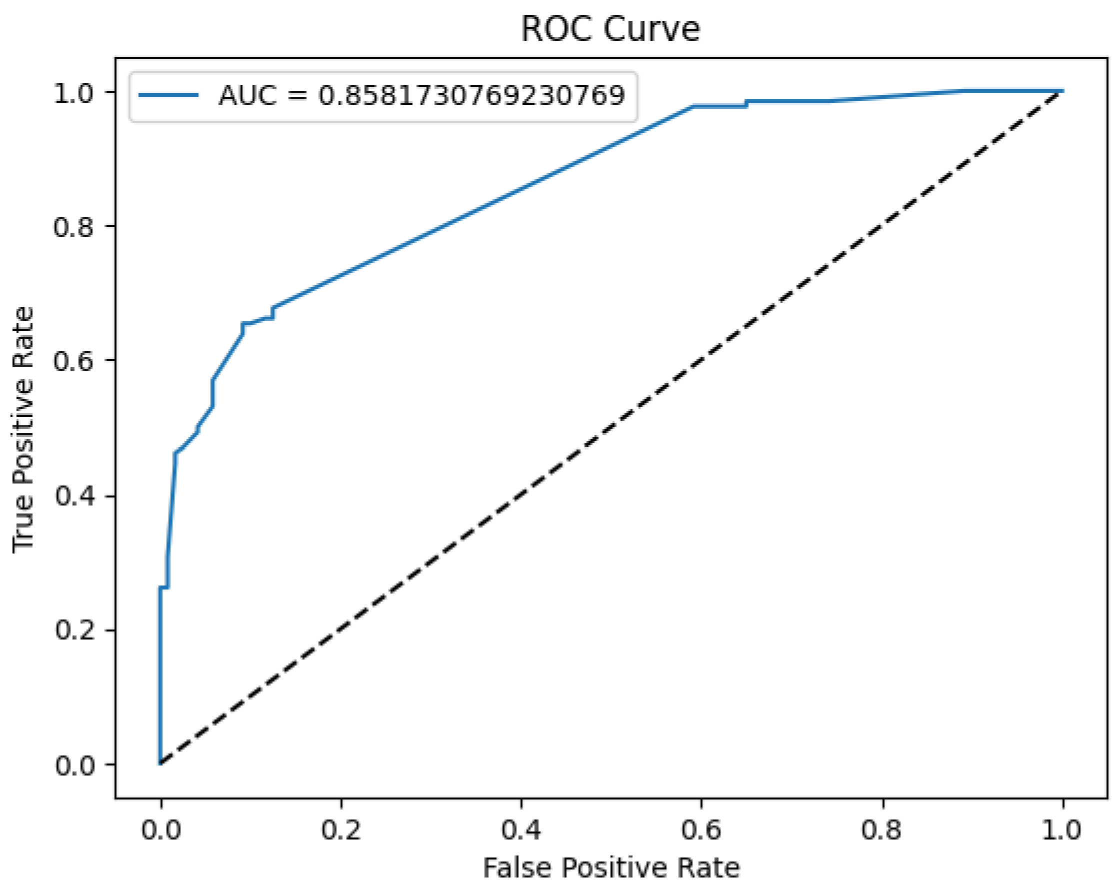
Result:

Figure 7: *Result of AdaBoost Classifier*



```
Best Parameters: {'algorithm': 'SAMME', 'learning_rate': 1.0, 'n_estimators': 200}

Confusion Matrix:
 [[106  14]
 [ 44  86]]
Classification Report:
               precision    recall  f1-score   support

           0       0.71      0.88      0.79       120
           1       0.86      0.66      0.75       130

    accuracy                           0.77       250
   macro avg       0.78      0.77      0.77       250
weighted avg       0.79      0.77      0.77       250

Accuracy: 0.768
AUC: 0.8581730769230769
```

Figure 8: *Receiver Operating Charateristics*



Analysis of the Result:

1. **Best Hyperparameters**

Table 24: *Table Showcasing Best Hyperparameters*

Hyperparameter	Description	Best Value
n_estimators	Number of boosting rounds	200
learning_rate	The rate at which the model learns, smaller values make the model learn slower	1.0
algorithm	The boosting algorithm: **SAMME** (discrete boosting) or **SAMME.R** (real boosting, weighted).	SAMME

2. **Analysis of the Confusion Matrix**

Table 25: *Table showing the interpretation of the confusion matrix*

Actual/ Predicted	**Predicted Label 0 (Negative Sentiment)**	**Predicted Label 1 (Positive Sentiment)**
Actual Label 0 (Negative Sentiment)	106 instances of label 0 are correctly classified (True Negatives)	14 instances of label 0 are misclassified as label 1 (False Positives)
Actual Label 1 (Positive Sentiment)	44 instances of label 1 are misclassified as label 0 (False Negatives)	86 instances of label 1 are correctly classified (True Positives)

3. **Analysis of Precision, Recall and F1-Score**

Table 26: *Table showing interpretation of the precision, recall and F1-Score of the two classes*

Target Labels (Label)	**Analysis**
0 (Negative Sentiment)	A precision of 0.71 suggests that when the model classifies label 0, it is correct 71% of the time. A recall of 0.88 suggests that the model is able to capture 88% of actual label 0 instances. An f-1 score of 0.79 suggests the overall balance between precision and recall. There are 120 actual instances of label 0 in total.
1 (Positive Sentiment)	A precision of 0.86 suggests that when the model classifies label 1, it is correct 86% of the time. A recall of 0.66 suggests that the model is able to capture 66% of actual label 1 instances. An f-1 score of 0.75 suggests the overall balance between precision and recall. There are 130 actual instances of label 1 in total.

4. **Analysis of Accuracy, Macro Average, and Weighted Average**

Table 27: *Table showing interpretation of Accuracy, Macro Average and Weighted Average of the trained model*

Accuracy	An accuracy of **76.8%** suggests that the model correctly classifies **76.8%** of the total instances.
Macro Average	This is the average of precision, recall, and F1-score across all classes, it is calculated without considering the support of each class. The macro average of **78%** for precision and

	77% for recall and f1 score suggests that on average, both the classes perform at a similar level.
Weighted Average	This is the average of precision, recall, and f1-score and it takes into account the support of each class. The weighted average in this case is **79%** for precision and **77%** for recall and f1-score, reflecting the overall performance weighted by class size.

5. **Analysis of the ROC Curve (Receiver Operating Characteristics)**

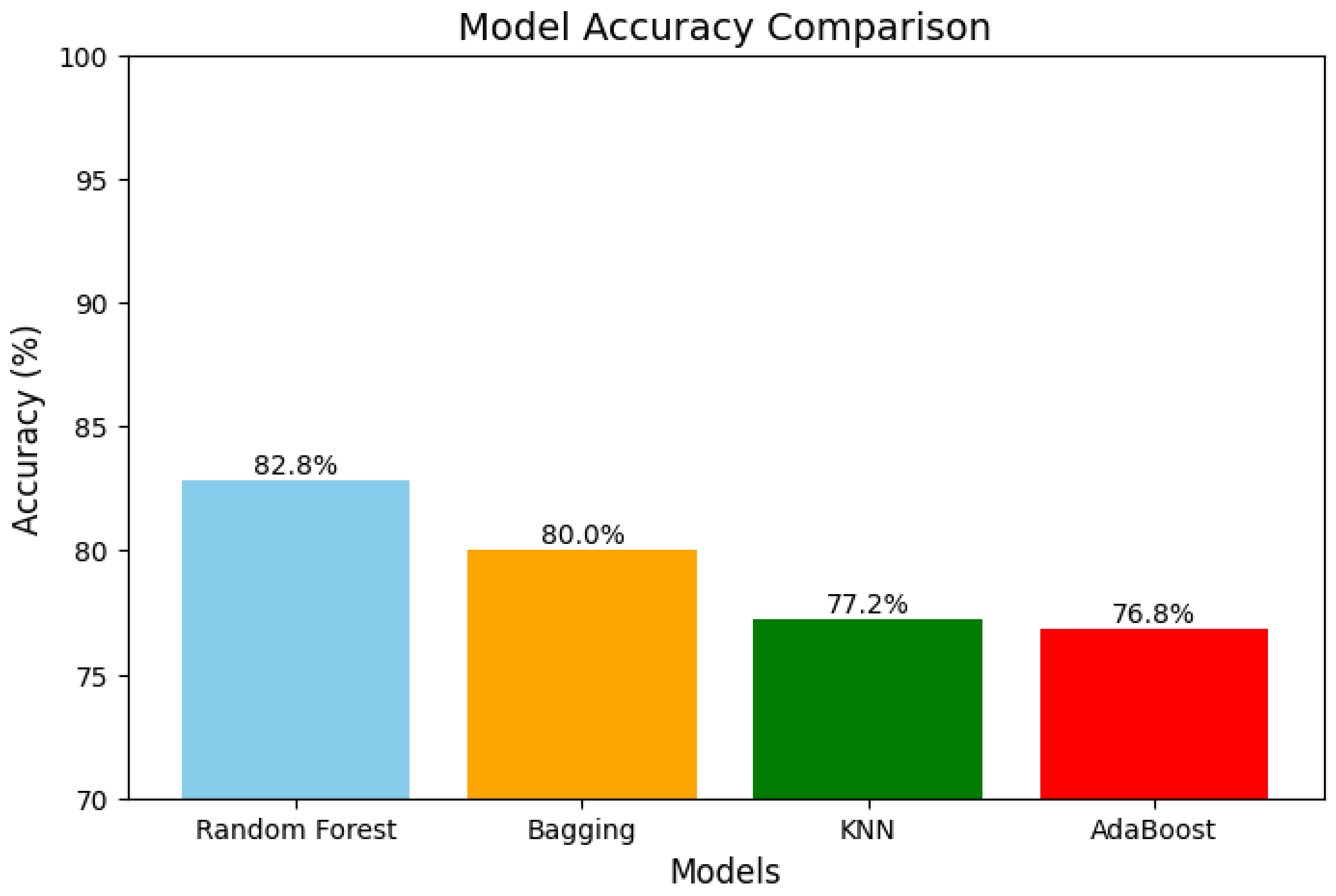
Table 28: *Table showing interpretation of the ROC Curve*

Aspect	**Observation**
AUC (Area Under the Curve)	The area under the ROC curve is 0.858 (AUC score). This score indicates that the model can effectively distinguish between the positive and negative class in our dataset.
Low False Positive Rate	The ROC curve remains close to the top-left which indicates minimum false positives
High True Positive Rate	The model effectively captures large number of true positive instances.
Above the Diagonal	The ROC curve lies significantly above the diagonal line which means that the model is not randomly guessing the outcomes.

2.3 Comparision and Analysis of the Classification Models

2.3.1 Visualization of the Classifiers and Ranking them on the basis of Accuracy

Figure 9: *Classifiers and their accuracies*



2.3.2 Analysis and Comparision

Table 29: *Table showcasing comparative analysis of the classifical models*

Classifier	**Accuracy (%)**	**Analysis**	**Rank**
Random Forest	82.8	It performs the best among all the classifiers with a highest accuracy of 82.8% which means this model correctly classifies 82.8% of overall instances.	1
Bagging	80	It performs slightly less than the Random Forest Classifier with an accuracy of 80%. This means that the model is able to correctly classify 80% of total instances.	2
KNN	77.2	It performs good but less than Bagging and Random Forest Classifier with a moderate accuracy of 77.2%. The KNN model is able to correctly classify 77.2% of total instances.	3
AdaBoost	76.8	This model ranks the lowest among all the classifiers with the least accuracy score of 76.8%. This model is only able to correctly predict 76.8% of overall instances.	4

Chapter 3. Classification using BERT and Roberta

3.1 Classification using BERT model

- The ‘Product_Review’ column in the ‘preprocessed_reviews’ DataFrame is the feature column, and the ‘Label’ column is the target column. These are split into 80% for training and 20% for testing where ‘train_texts’ contains 80% of the feature texts for training, ‘val_texts’ contains 20% of the feature texts for testing, ‘train_labels’ contains 80% of the target data for training, and ‘val_labels’ contains 20% of the target data for testing.
- BertTokenizer is used from the Transformers library to tokenize and enode the text data into numerical format making it suitable for input to a BERT model. The ‘encode_data’ function is used to processes the list of texts using the tokenizer, making the series padded, truncated to a maximum length of 128, and returned as PyTorch tensors. The ‘train_texts’ and ‘val_texts’ are encoded into ‘train_encodings’ and ‘val_encodings’ respectively.
- A **custom** PyTorch Dataset called ReviewsDataset, which is used to structure and prepare the training and validation data for the BERT model.
- BertForSequenceClassification model is imported from the Hugging Face Transformers library and the version is ‘bert-base-uncased’ which has 12 hidden layers and is trained on 110 million parameters.
- The following configurations are used to train the model:

Table 30: *Table showing different configurations used*

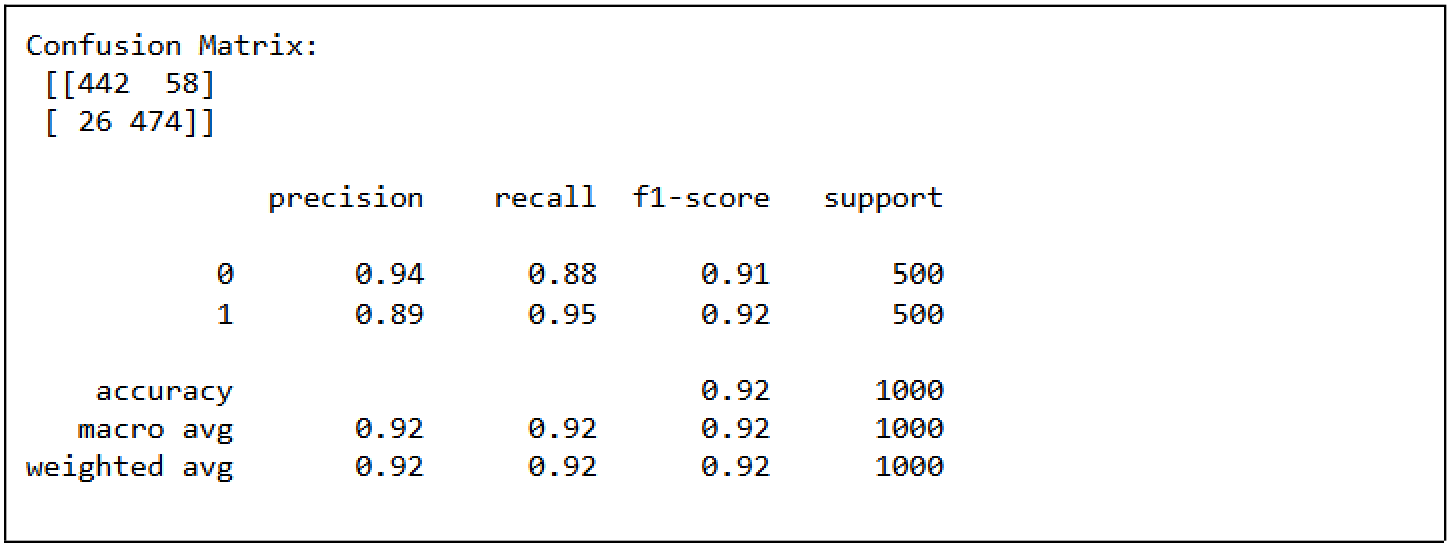
Parameter	**Description**	**Value**
output_dir	Directory for saving model checkpoints and results	‘./results’
run_name	A unique name given for this training	“bert_sentiment_analysis”
num_train_epochs	It is the number of training epochs	3
per_device_train_batch_size	It is the batch size for training	16
per_device_eval_batch_size	It is the batch size for evaluation	64
warmup_steps	Warmup steps for learning rate scheduler	500
weight_decay	Weight decay for regularization	0.01
logging_dir	Directory for storing logs	‘./logs’
evaluation_strategy	To evaluate at the end of each epoch	“epoch”
save_strategy	To save the model checkpoint at each	“epoch”

	epoch	
logging_steps	To log training metrics every specified number of steps	50
load_best_model_at_the_end	To load the best model based on validation loss	True
save_total_limit	To keep only the last specified number of scheckpoints	2

- The trainer class is initialized to manage the training and evaluation process.
- The model is trained based on the set parameters.
- The performance of the model is evaluated using a confusion matrix, classification report, accuracy score, and AUC (area under the curve) score

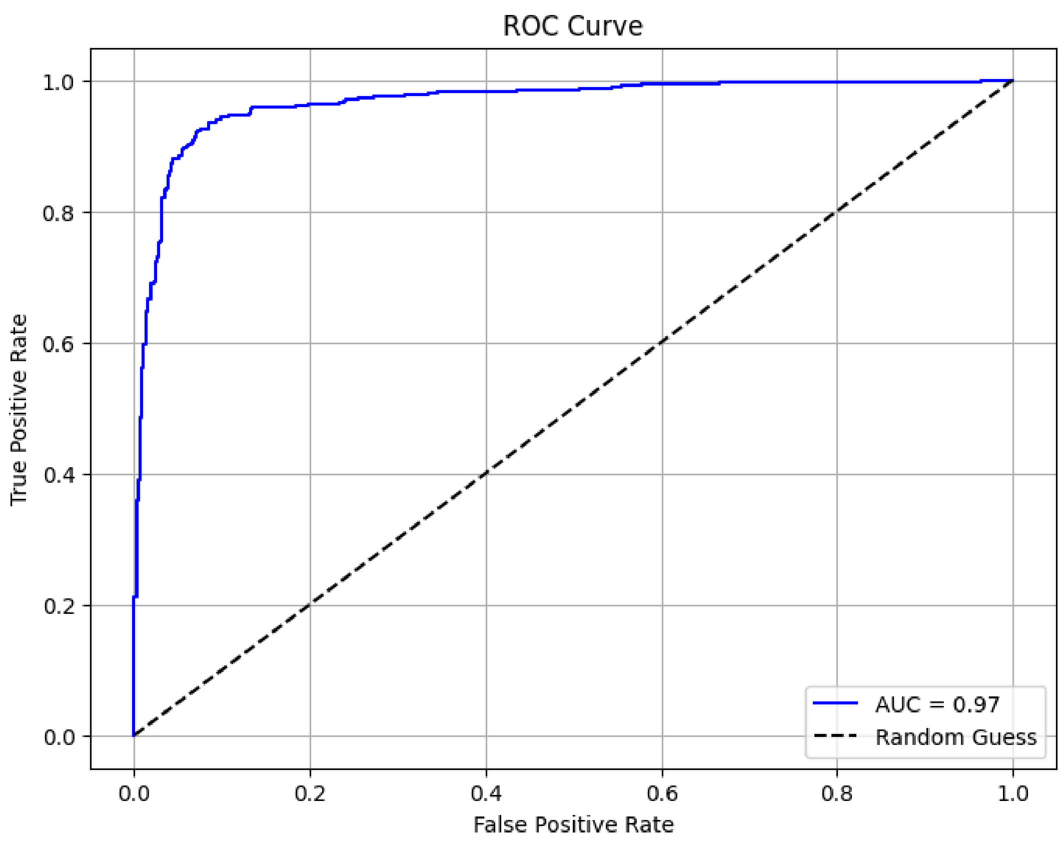
Result:

Figure 10: *Result of BERT fine-tuned classification model*



```
Confusion Matrix:
 [[442  58]
 [ 26 474]]

              precision    recall  f1-score   support

           0       0.94      0.88      0.91       500
           1       0.89      0.95      0.92       500

    accuracy                           0.92      1000
   macro avg       0.92      0.92      0.92      1000
weighted avg       0.92      0.92      0.92      1000
```

Figure 11: *Receiver Operating Characteristics*



Analysis of the Result

1. **Analysis of the Confusion Matrix**

Table 31: *Table showing interpretation of the confusion matrix*

Actual/ Predicted	**Predicted Label 0 (Negative Sentiment)**	**Predicted Label 1 (Positive Sentiment)**
Actual Label 0 (Negative Sentiment)	442 instances of label 0 are correctly classified (True Negatives)	58 instances of label 0 are misclassified as label 1 (False Positives)
Actual Label 1 (Positive Sentiment)	26 instances of label 1 are misclassified as label 0 (False Negatives)	474 instances of label 1 are correctly classified (True Positives)

2. Analysis of Precision, Recall and F1-Score

Table 32: *Table showing interpretation of the precision, recall and F1-Score of the two classes*

Target Labels (Label)	Analysis
0 (Negative Sentiment)	A precision of 0.94 suggests that when the model classifies label 0, it is correct 94% of the time. A recall of 0.88 suggests that the model is able to capture 88% of actual label 0 instances. An f-1 score of 0.91 suggests the overall balance between precision and recall. There are 500 actual instances of label 0 in total.
1 (Positive Sentiment)	A precision of 0.89 suggests that when the model classifies label 1, it is correct 89% of the time. A recall of 0.95 suggests that the model is able to capture 95% of actual label 1 instances. An f-1 score of 0.92 suggests the overall balance between precision and recall. There are 500 actual instances of label 1 in total.

3. Analysis of Accuracy, Macro Average, and Weighted Average

Table 33: *Table showing interpretation of Accuracy, Macro Average and Weighted Average of the trained model*

Accuracy	An accuracy of **92%** suggests that the model correctly classifies **92%** of the total instances.
Macro Average	This is the average of precision, recall, and F1-score across all classes, it is calculated without considering the support of each class. The macro average of **92%**, for precision, recall and f1 score suggests that on average, both the classes perform at a similar level.
Weighted Average	This is the average of precision, recall, and F1-score and it takes into account the support of each class. The weighted average in this case is **92%**, reflecting the overall performance weighted by class size.

4. **Analysis of the ROC Curve (Receiver Operating Characteristics)**

Table 34: *Table showing interpretation of the ROC Curve*

Aspect	Observation
AUC (Area Under the Curve)	The area under the ROC curve is 0.97 (AUC score). This high score indicates that the model can effectively distinguish between the positive and negative class in our dataset.
Low False Positive Rate	The ROC curve remains close to the top-left which indicates minimum false positives
High True Positive Rate	The model effectively captures large number of true positive instances.
Above the Diagonal	The ROC curve lies significantly above the diagonal line which means that the model is not randomly guessing the outcomes.

3.2 Classification using RoBERTa Model

- RobertaTokenizer is used from the Transformers library to tokenize and enode the text data into numerical format making it suitable for input to a Roberta model. The 'encode_data' function is used to processes the list of texts using the tokenizer, making the series padded, truncated to a maximum length of 128, and returned as PyTorch tensors. The 'train_texts' and 'val_texts' are encoded into 'train_encodings' and 'val_encodings' respectively.
- A custom PyTorch Dataset called ReviewsDataset, which is used to structure and prepare the training and validation data for the Roberta model.
- RobertaForSequenceClassification model is imported from the Hugging Face Transformers library, and the version is roberta-base, which has 12 hidden layers and is trained on 125 million parameters.
- The following configurations are used to train the model:

Table 35: *Table showing different configurations used*

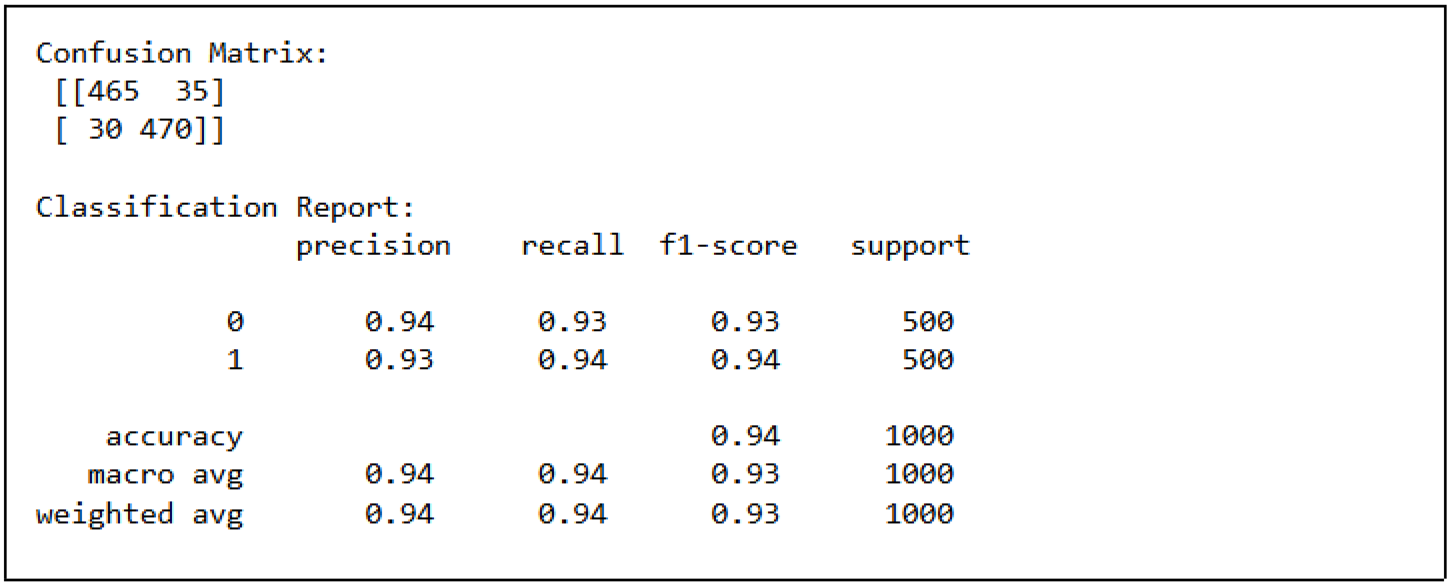
Parameter	Description	Value
output_dir	Directory for saving model checkpoints and results	'./results'
run_name	A unique name given for this training	"roberta_sentiment_analysis"
num_train_epochs	It is the number of training epochs	3

per_device_train_batch_size	It is the batch size for training	16
per_device_eval_batch_size	It is the batch size for evaluation	64
warmup_steps	Warmup steps for learning rate scheduler	500
weight_decay	Weight decay for regularization	0.01
logging_dir	Directory for storing logs	'./logs'
evaluation_strategy	To evaluate at the end of each epoch	"epoch"
save_strategy	To save the model checkpoint at each epoch	"epoch"
logging_steps	To log training metrics every specified number of steps	50
load_best_model_at_the_end	To load the best model based on validation loss	True
save_total_limit	To keep only the last specified number of scheckpoints	2

- The Trainer class is initialized to manage the training and evaluation process.
- The model is trained on the specified parameters and datasets using the trainer.train() method

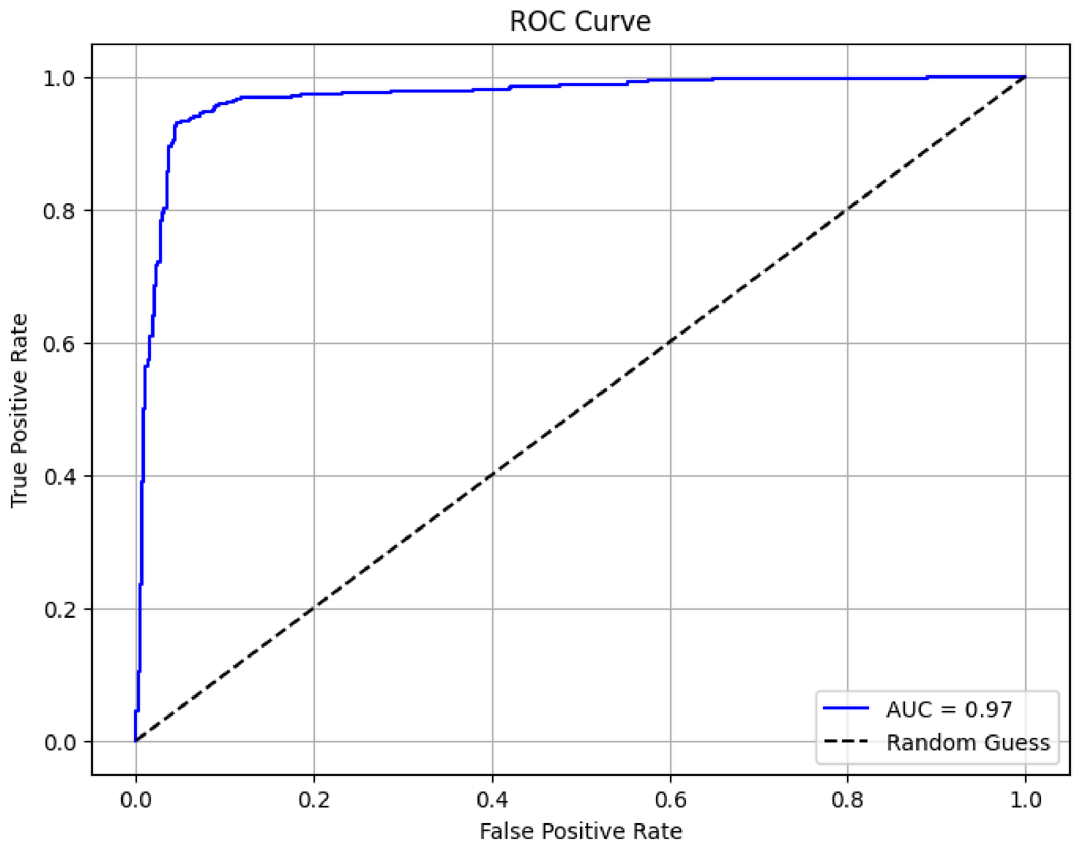
Result:

Figure 12: *Result of RoBERTa fine-tuned classification model*



```
Confusion Matrix:
 [[465  35]
 [ 30 470]]

Classification Report:
               precision    recall  f1-score   support

           0       0.94      0.93      0.93       500
           1       0.93      0.94      0.94       500

    accuracy                           0.94      1000
   macro avg       0.94      0.94      0.93      1000
weighted avg       0.94      0.94      0.93      1000
```

Figure 13: *Receiver Operating Characteristics*



Analysis of the Result

1. **Analysis of the Confusion Matrix**

Table 36: *Table showing interpretation of the confusion matrix*

Actual/ Predicted	**Predicted Label 0 (Negative Sentiment)**	**Predicted Label 1 (Positive Sentiment)**
Actual Label 0 (Negative Sentiment)	465 instances of label 0 are correctly classified (True Negatives)	35 instances of label 0 are misclassified as label 1 (False Positives)
Actual Label 1 (Positive Sentiment)	30 instances of label 1 are misclassified as label 0 (False Negatives)	470 instances of label 1 are correctly classified (True Positives)

2. **Analysis of Precision, Recall and F1-Score**

Table 37: *Table showing interpretation of the precision, recall and F1-Score of the two classes*

Target Labels (Label)	**Analysis**
0 (Negative Sentiment)	A precision of 0.94 suggests that when the model classifies label 0, it is correct 94% of the time. A recall of 0.93 suggests that the model is able to capture 93% of actual label 0 instances. An f-1 score of 0.93 suggests the overall balance between precision and recall. There are 500 actual instances of label 0 in total.
1 (Positive Sentiment)	A precision of 0.93 suggests that when the model classifies label 1, it is correct 93% of the time. A recall of 0.94 suggests that the model is able to capture 94% of actual label 1 instances. An f-1 score of 0.94 suggests the overall balance between precision and recall. There are 500 actual instances of label 1 in total.

3. **Analysis of Accuracy, Macro Average, and Weighted Average**

Table 38: *Table showing interpretation of Accuracy, Macro Average and Weighted Average of the trained model*

Accuracy	An accuracy of **94%** suggests that the model correctly classifies **94%** of the total instances.
Macro Average	This is the average of precision, recall, and F1-score across all classes, it is calculated without considering the support of each class. The macro average of **94%** for precision and recall and **93 %** for f1 score suggests that on average, both the classes perform at a similar level.
Weighted Average	This is the average of precision, recall, and F1-score and it takes into account the support of each class. The weighted average in this case is **94%** for precision and recall and **93%** for f1-score, reflecting the overall performance weighted by class size.

4. **Analysis of the ROC Curve (Receiver Operating Characteristics)**

Table 39: *Table showing interpretation of the ROC Curve*

Aspect	**Observation**
AUC (Area Under the Curve)	The area under the ROC curve is 0.97 (AUC score). This high score indicates that the model can effectively distinguish between the positive and negative class in our dataset.
Low False Positive Rate	The ROC curve remains close to the top-left which indicates minimum false positives
High True Positive Rate	The model effectively captures large number of true positive instances.
Above the Diagonal	The ROC curve lies significantly above the diagonal line which means that the model is not randomly guessing the outcomes.

3.3 Comparision and Analysis between BERT and RoBERTa

Table 40: *Table for comparative analysis between BERT and RoBERTa*

Classifier	**Accuracy (%)**	**Analysis**	**Rank**
RoBERTa	94	It performs better than BERT model with an excellent accuracy score of 94%. This implies that the BERT model is able to correctly classify 94% of total instances.	1
BERT	92	It performs slightly less than the RoBERTa model yet achieves a very good accuracy score of 92% which implies that the model is able to correctly predict 92% of overall instances.	2

3.4 Comparision and Analysis of BERT and RoBERTa Model with Algorithms Used in Chapter 2

Figure 14: *All classifiers and their accuracies*

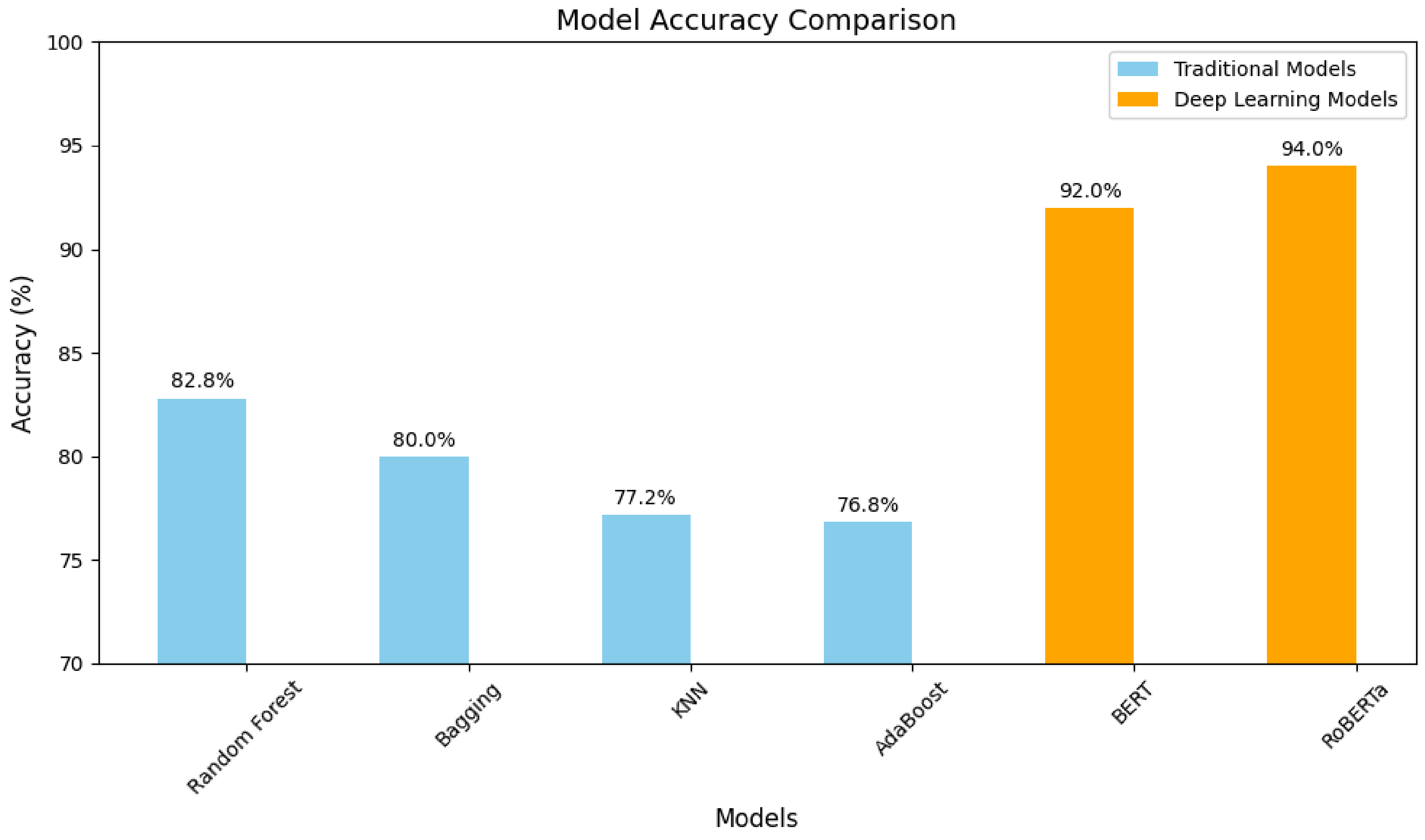


Table 41: *Table showcasing comparative analysis between BERT and RoBERTa with models from chapter 2*

Classifier	**Accuracy (%)**	**Analysis**
RoBERTa	94	From the above figure, it is very clear that the **RoBERTa** model significantly performs better than traditional classification models such as **Random Forest**, **Bagging**, **KNN**, and **AdaBoost**, achieving an exceptional accuracy of **94%** compared to Random Forest (**82.8%**), Bagging (**80%**), KNN (**77.2%**), and AdaBoost (**76.8%**). This outstanding performance is because of RoBERTa's transformer-based architecture, specifically the RoBERTa-base variant, which consists of 12 hidden layers and is trained on approximately 125 million parameters. These advanced features enable RoBERTa to effectively capture complex patterns and relationships in text which makes the model very suitable for text classification.
BERT	92	The **BERT** model performs slightly below the **RoBERTa** model but still achieves a very good accuracy score of 92%

		and surpasses traditional models such as **Random Forest (82.8%)**, **Bagging(80%)**, **KNN(77.2%)**, and **AdaBoost(76.8%)**. This strong performance is due to BERT's transformer-based architecture, specifically the BERT-base variant, which consists of 12 hidden layers and is trained on approximately 110 million parameters. These features enable BERT to classify texts in far more better way than the traditional models.

Chapter 4. Topic Detection

4.1 Topic Detection using LDA

- The ‘Product_Review’ column in the ‘preprocessed_reviews’ DataFrame is tokenized making it suitable to perform topic detection.
- The tokenized texts are converted into a list.
- A gensim dictionary (id2word) is created from the tokenized words where each unique word is assigned an ID.
- Rare words (words appearing in fewer than 5 documents) and most frequent words (words which are present in more than 50% of the documents) are removed.
- The text data is converted into a bag-of-words (BoW) representation.
- An LDA model is trained to retrieve 10 topics.
- The topics are visualized using the pyLDAvis library

Result:

Figure 15: *Topic 1*

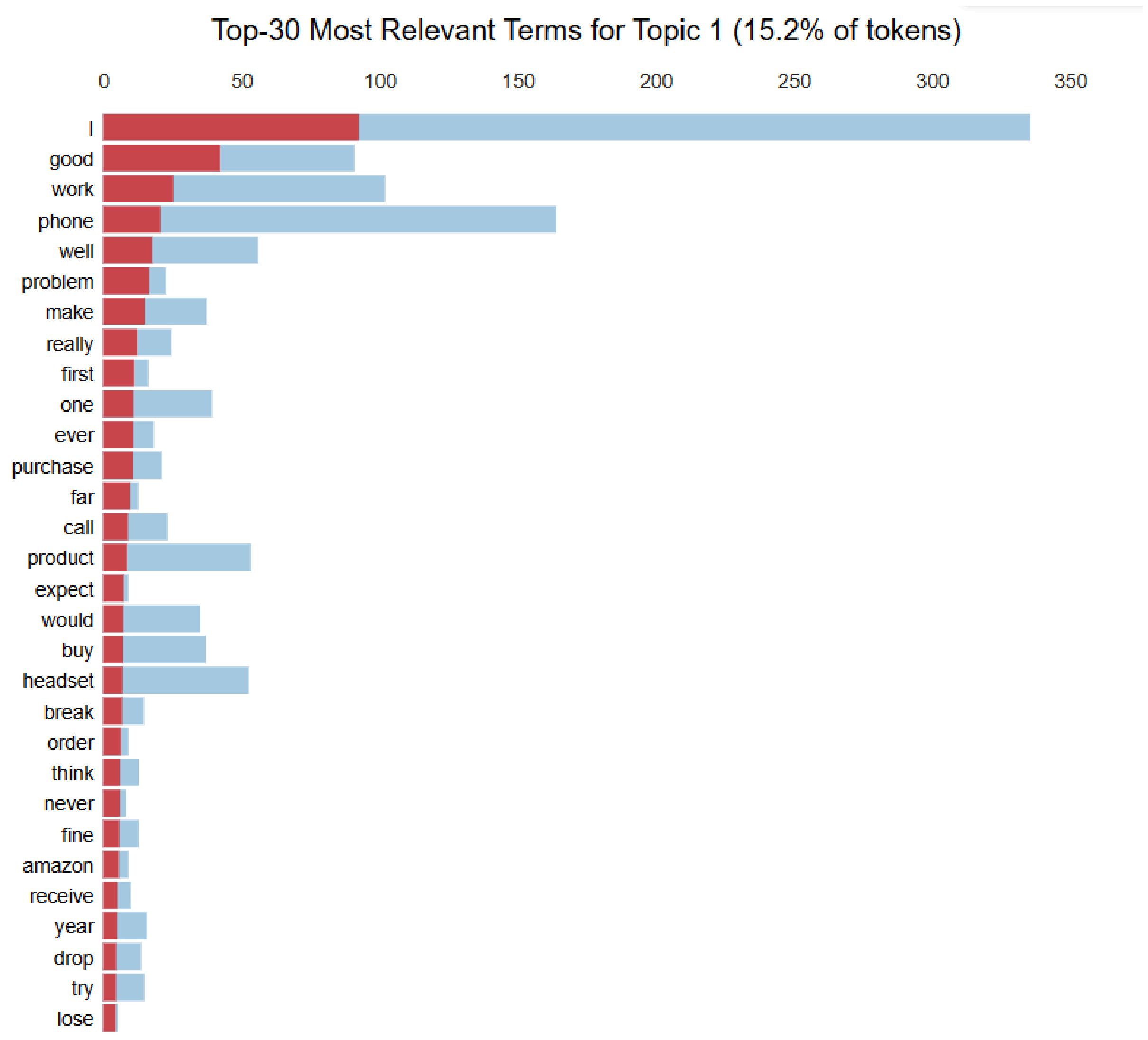


Figure 16: *Topic 2*

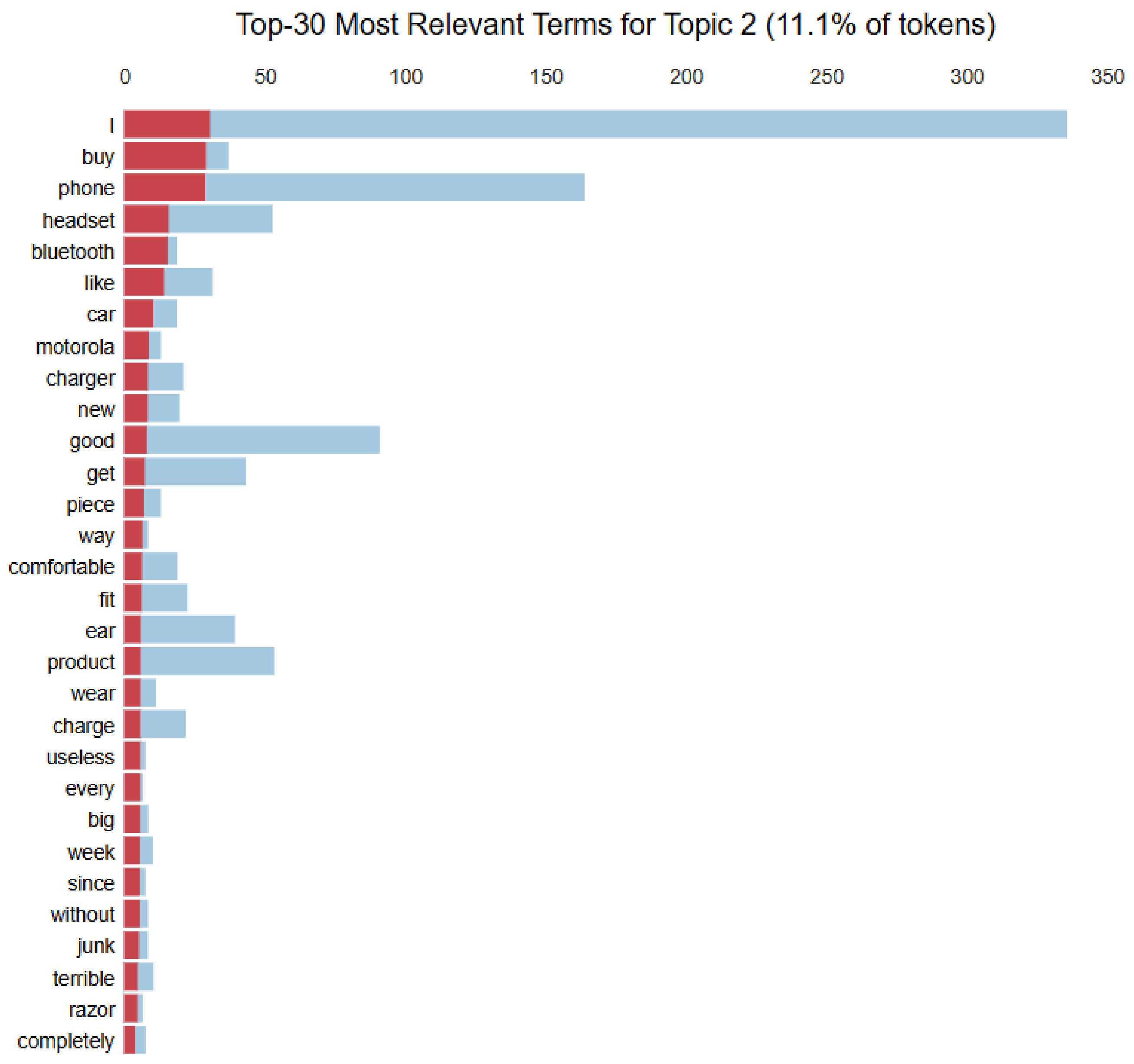


Figure 17: *Topic 3*

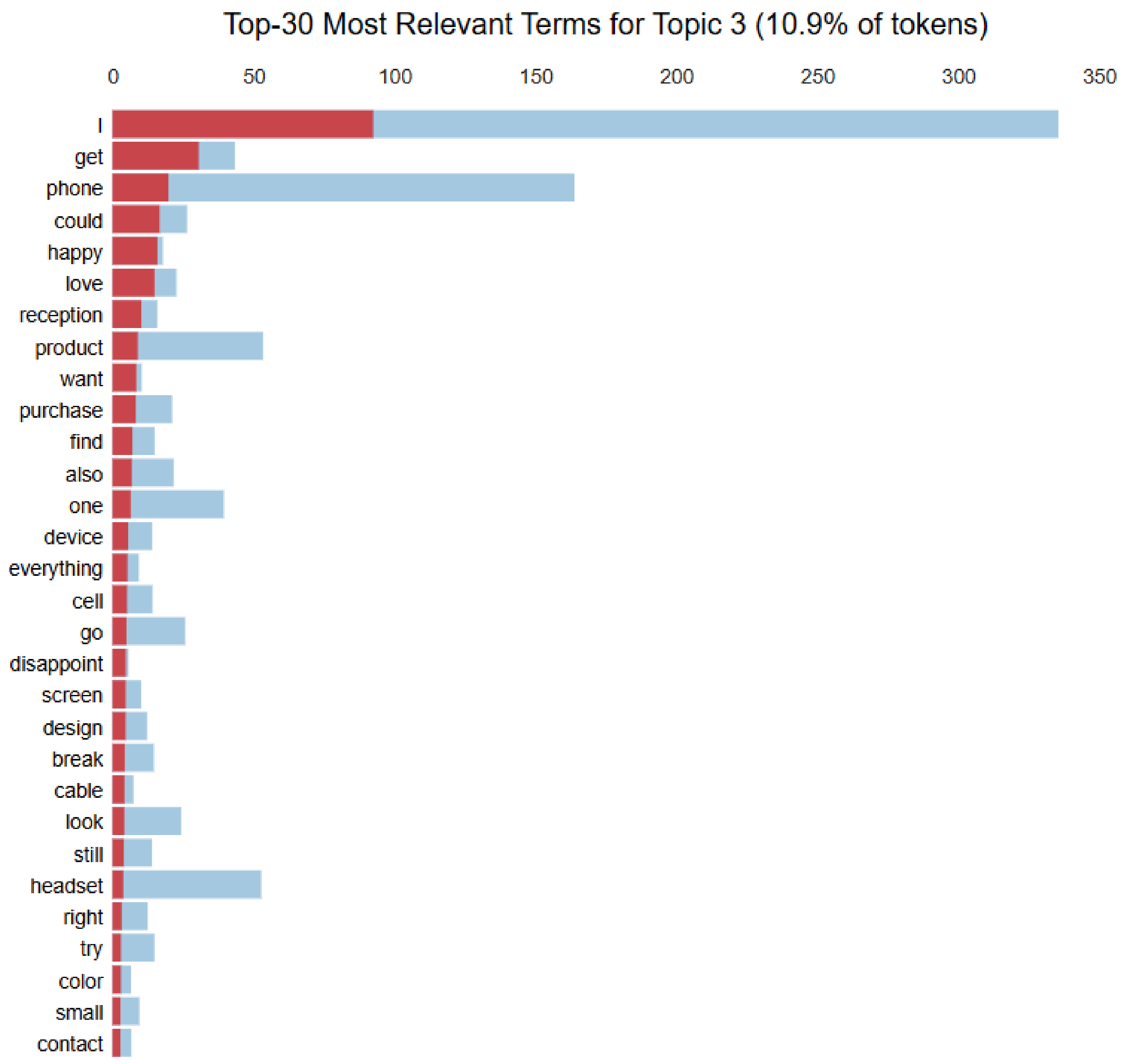


Figure 18: *Topic 4*

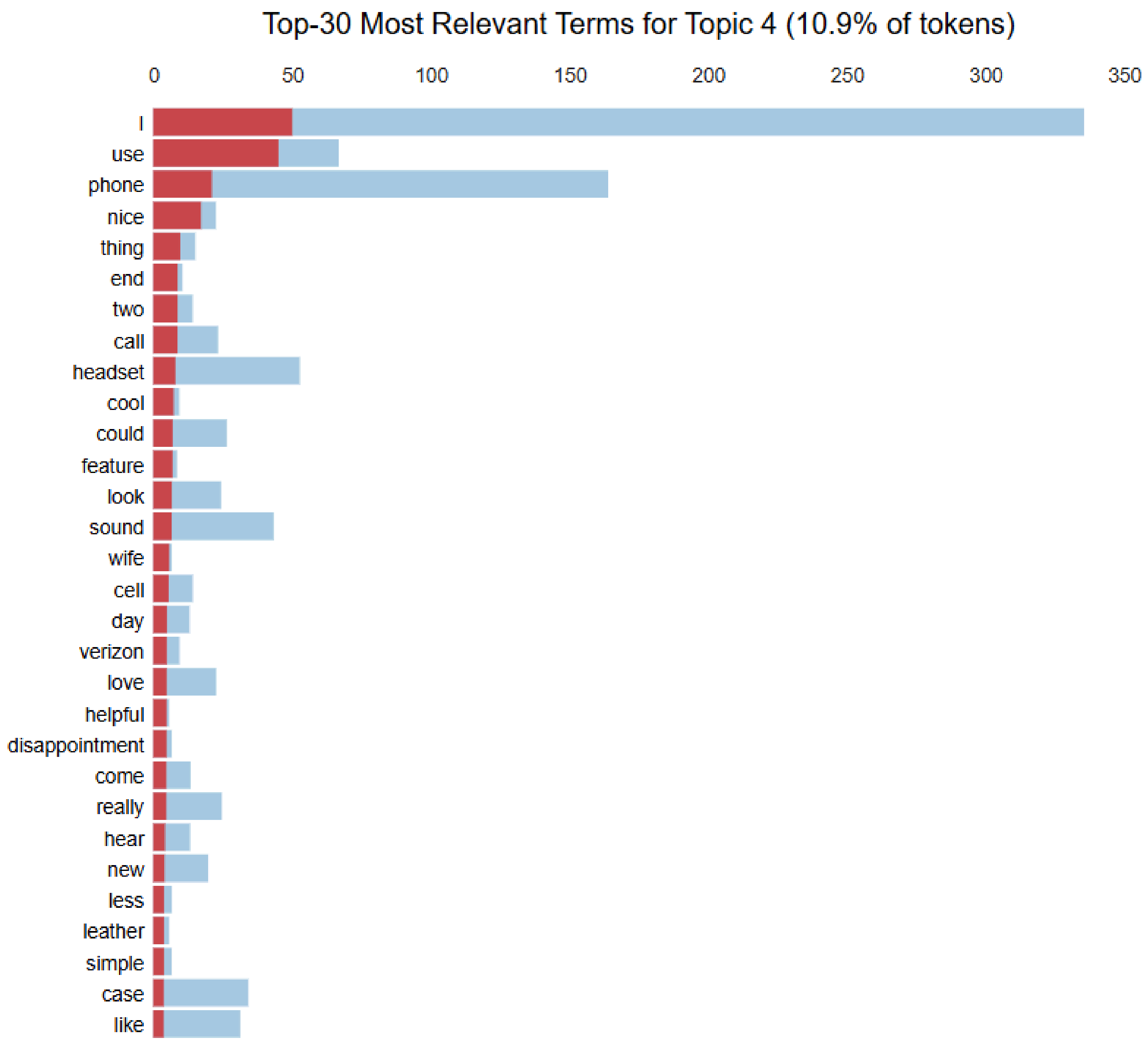


Figure 19: *Topic 5*

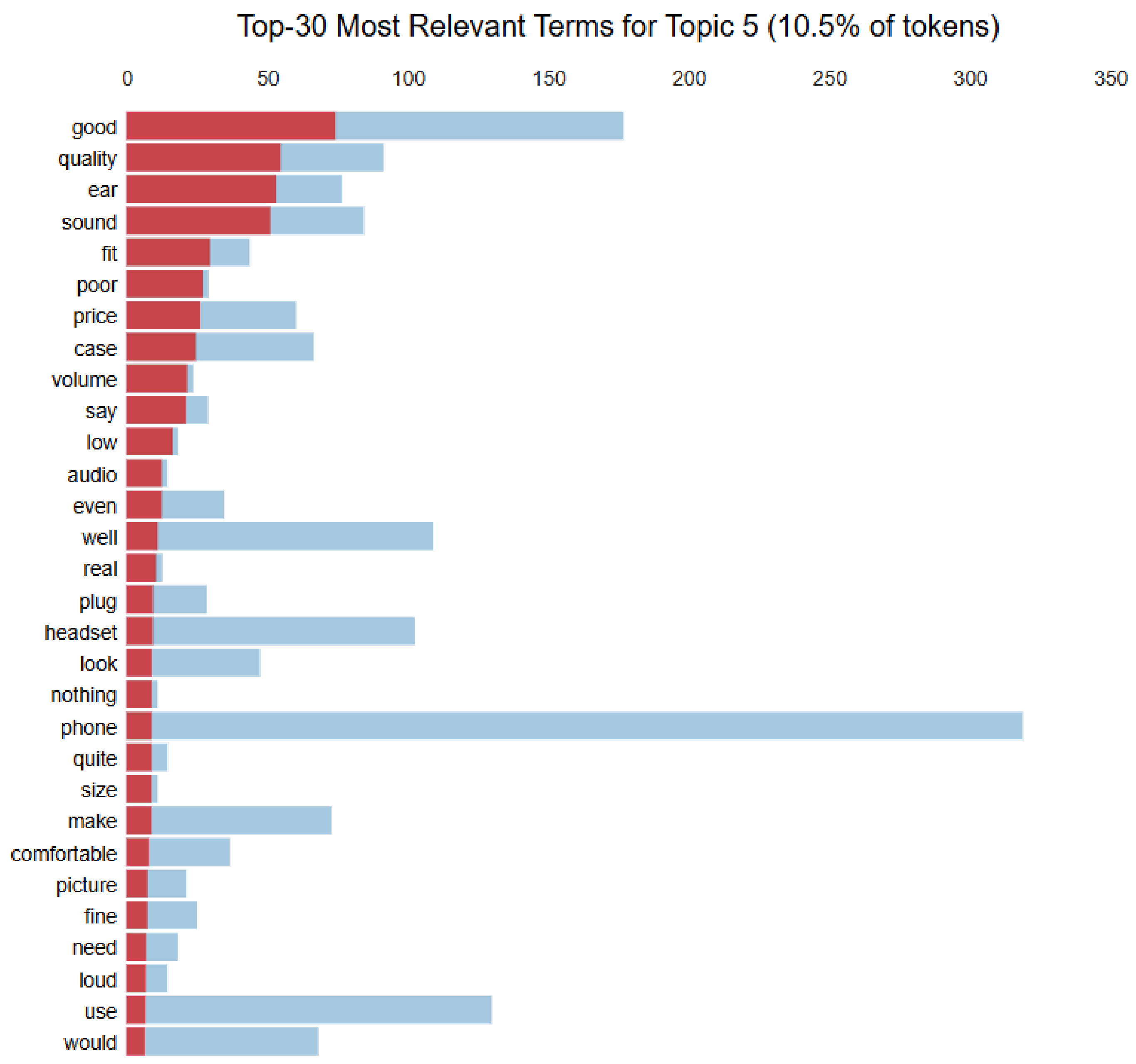


Figure 20: *Topic 6*

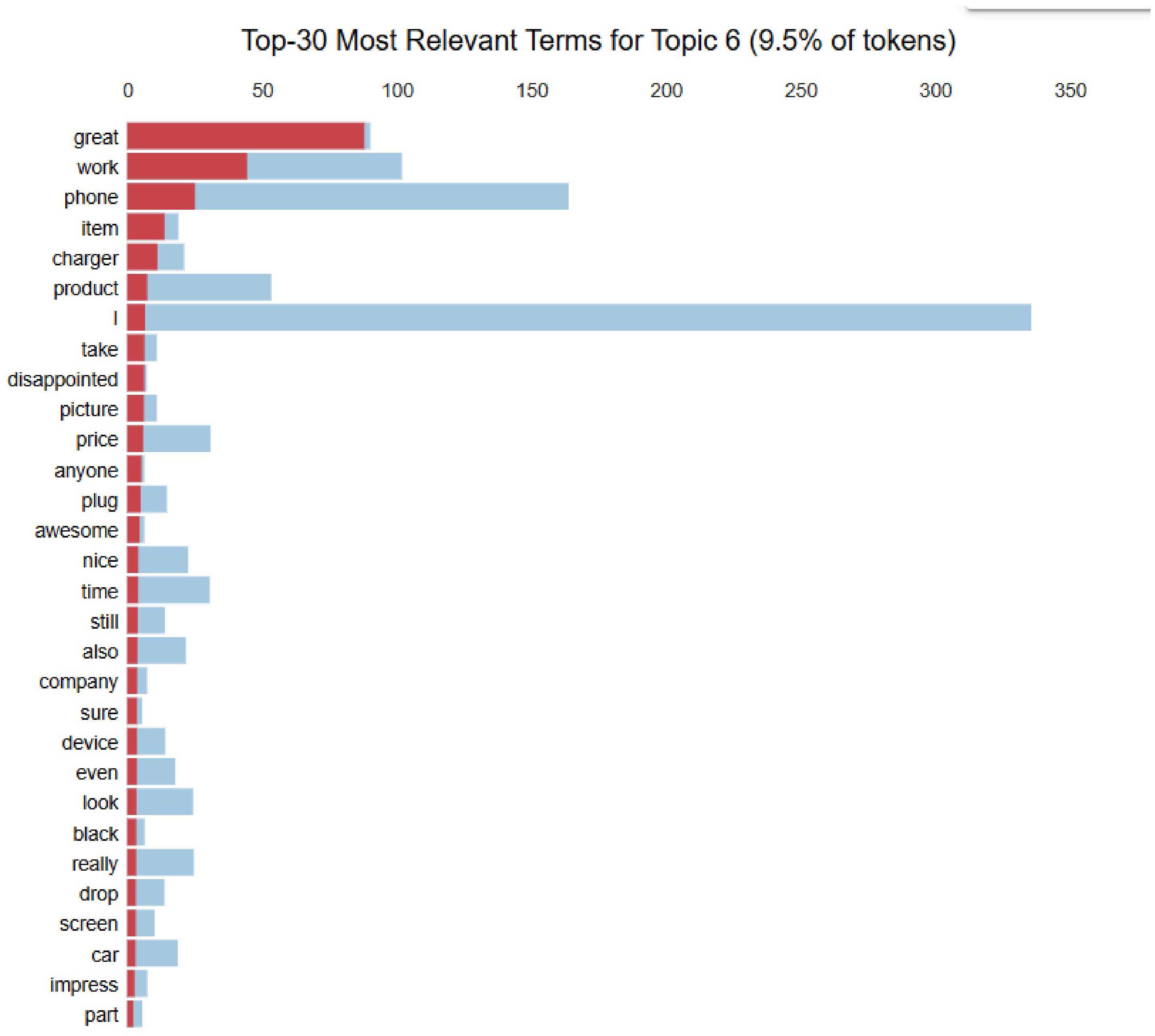


Figure 21: *Topic 7*

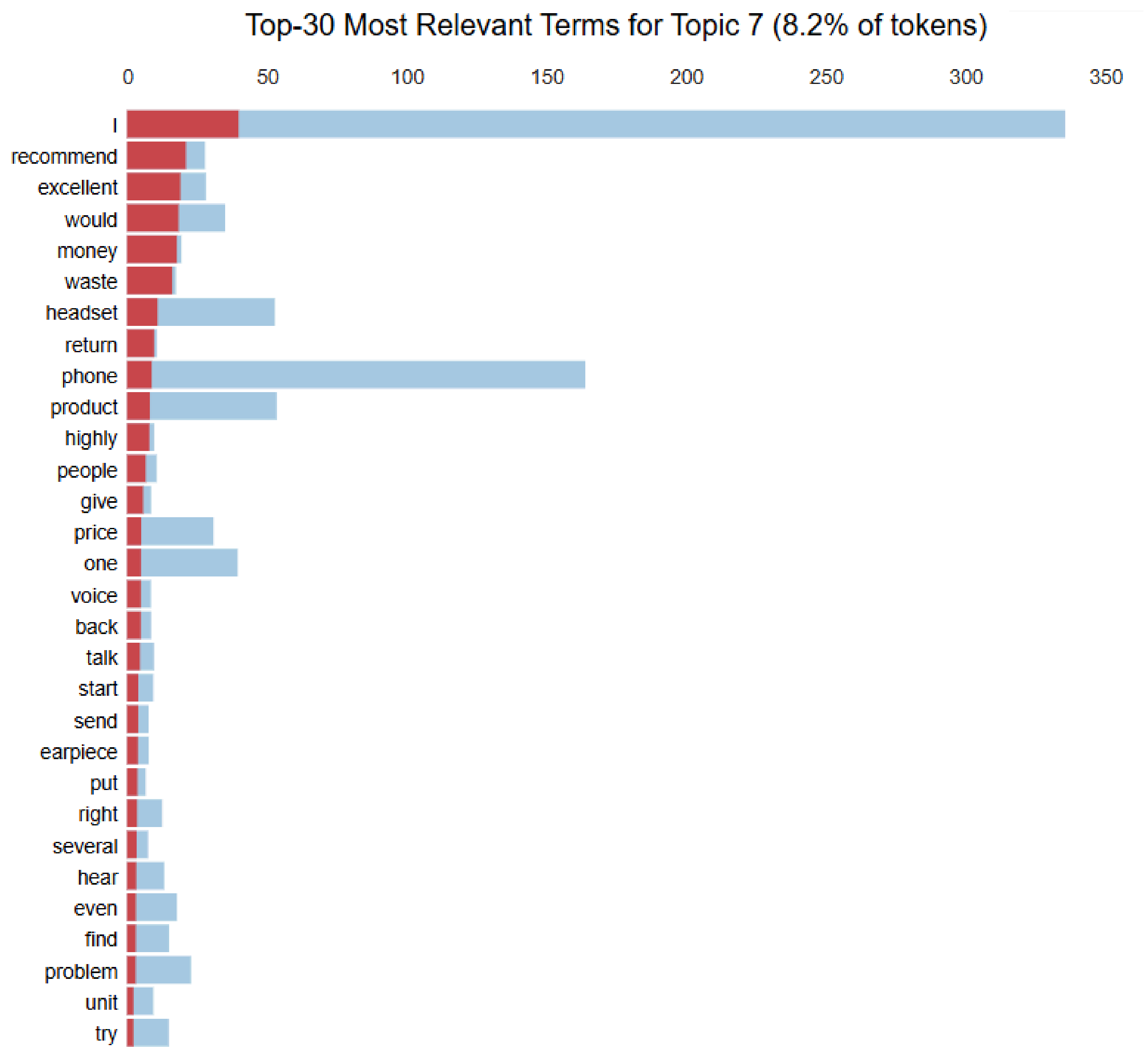


Figure 22: *Topic 8*

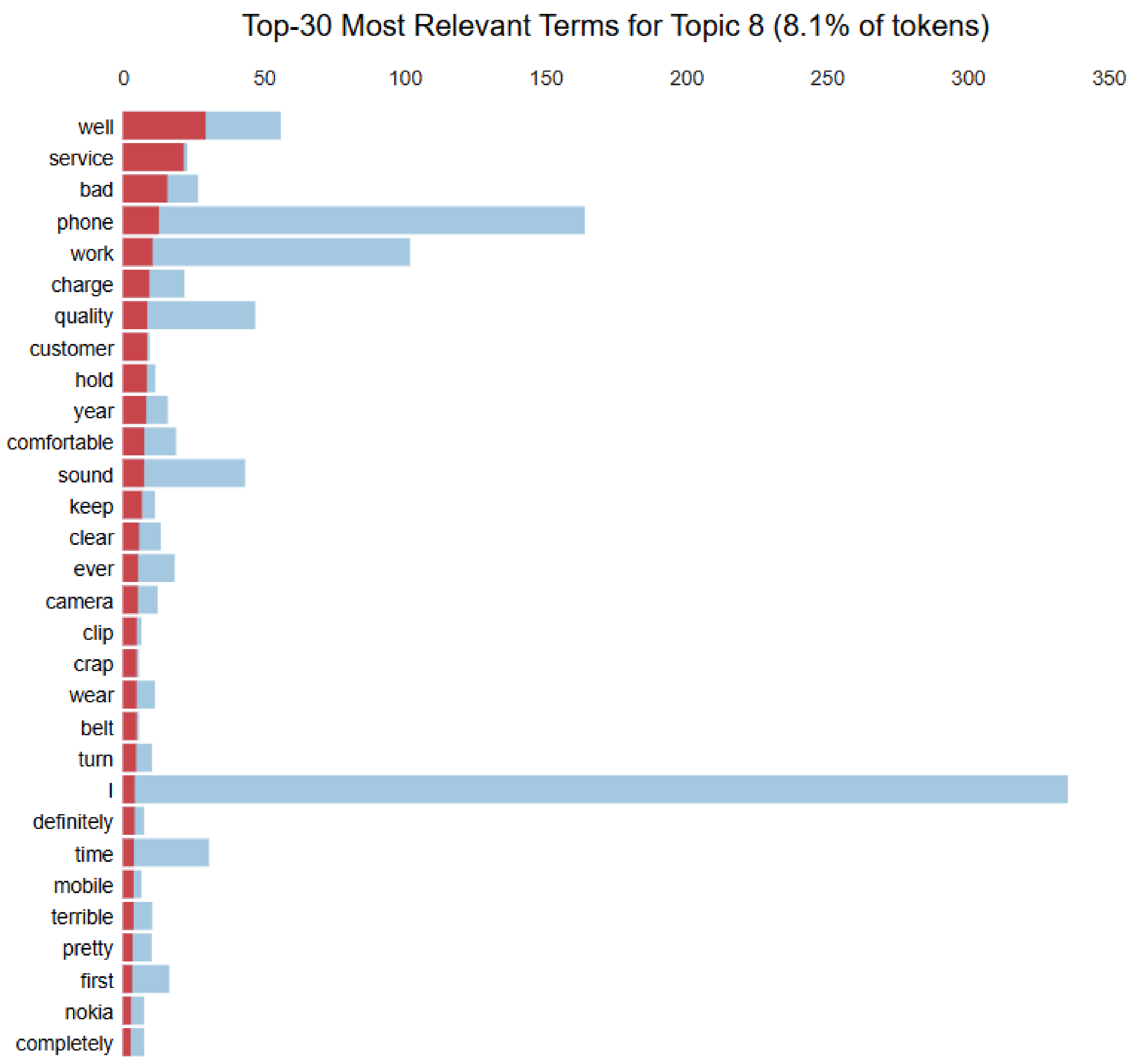


Figure 23: *Topic 9*

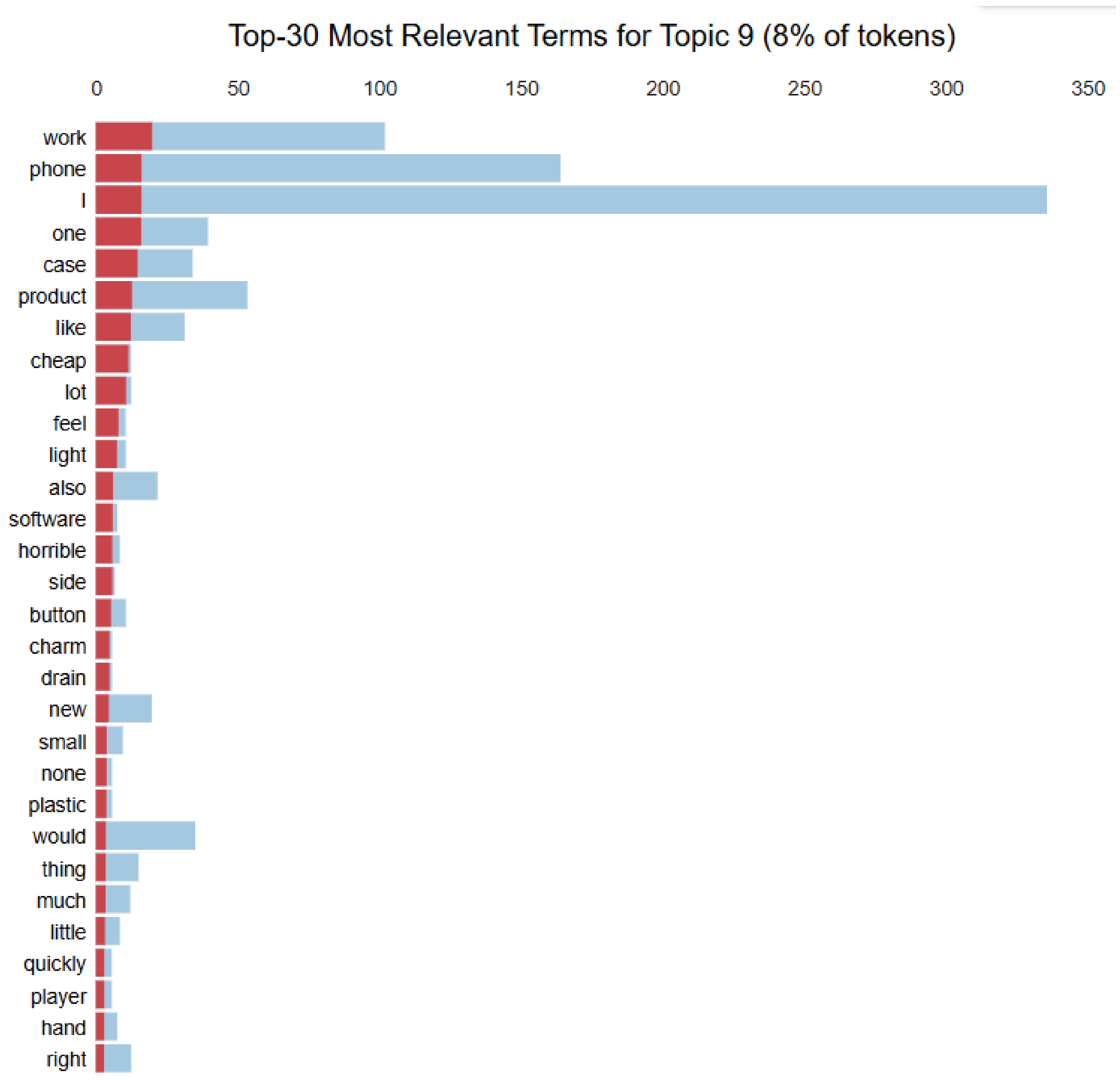
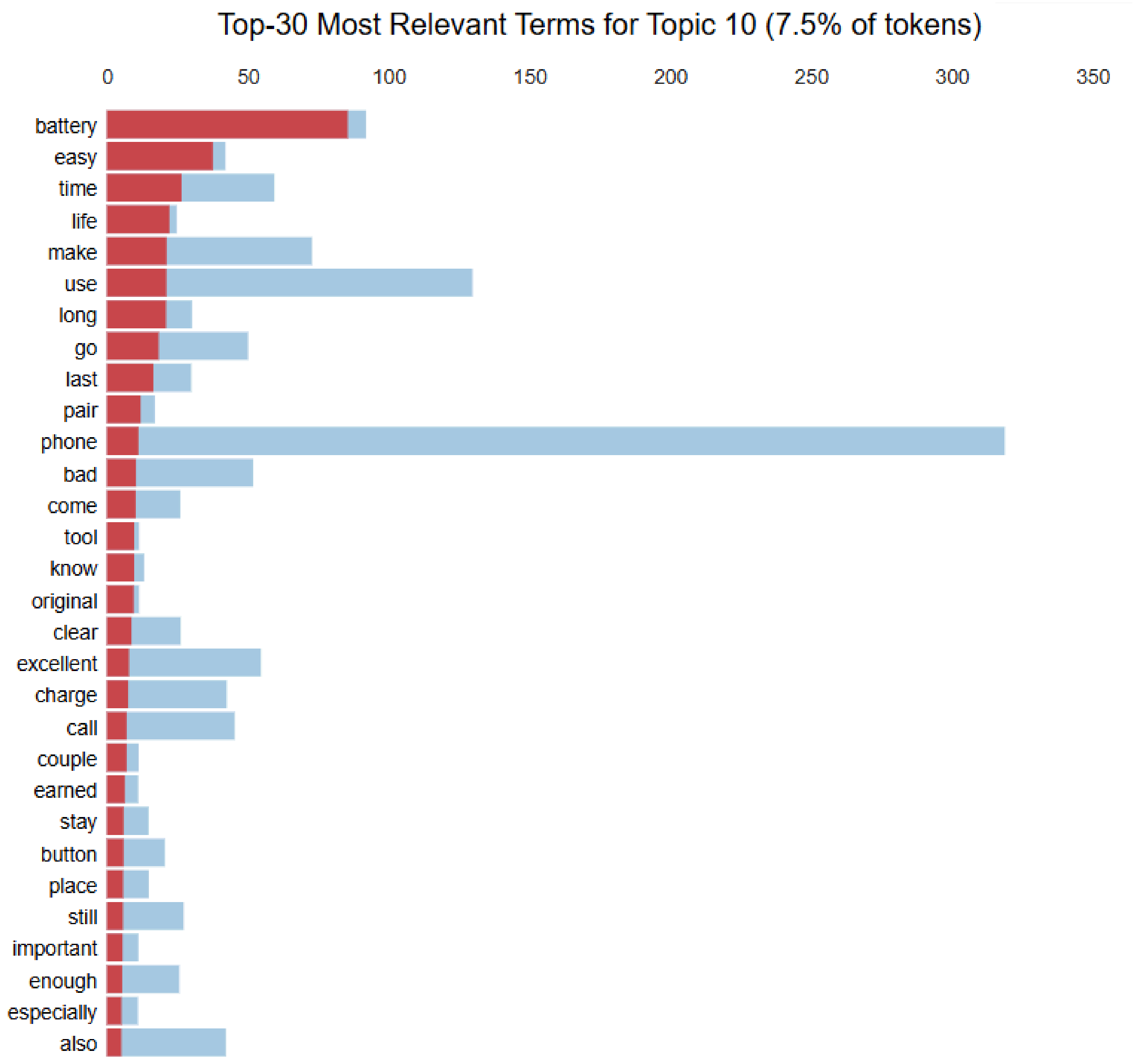


Figure 24: *Topic 10*



Interpretation of the Detected Topics

Table 42: *Topic wise interpretation of the detected topics*

Topic No.	**Words Present**	**Interpretation of Topic and Quality Assessment**
1	"I", "good", "work", "phone", "well", "problem", "make", "really", "first", "one", "ever", "purchase", "far", "call", "product", "expect", "would", "buy", "headset", "break", "order", "think", "never", "fine", "amazon", "receive", "year", "drop", "try", "lose"	**Interpretation:** This topic is about reviews of electronics, such as phones and headsets, referring to their performance, quality, and purchasing experience, including satisfaction with online orders via Amazon. The topic shows both positive and negative aspects, for example, "good,", "fine," and possible issues such as "problem," "break," "drop", "lose" **Quality:** This topic holds **15.2%** of total tokens (words) in the dataset with the most frequent term as 'I' and the least frequently occurring terms are 'year', 'drop', 'try', 'lose'
2	"I", "buy", "phone", "headset", "bluetooth", "like", "car", "motorola", "charger", "new", "good", "get", "piece", "way", "comfortable", "fit", "ear", "product", "wear", "charge", "useless", "every", "big", "week", "since", "without", "junk", "terrible", "razor", "completely"	**Interpretation:** This topic outlines customer feedback about Bluetooth headsets, mainly Motorola devices, focusing on their comfort, fit, and use in cars. While some users were satisfied when it comes to features like comfort and design, others raised issues with quality, battery life, and reliability, describing some products as "useless" or "junk." **Quality:** This topic holds **11.1%** of total tokens (words) in the dataset with the most frequent term as 'I'.
3	"I", "get", "phone", "could", "happy", "love", "reception",	**Interpretation:** This topic summarizes customers'

	"product", "want", "purchase", "find", "also", "one", "device", "everything", "cell", "go", "disappoint", "screen", "design", "break", "cable", "look", "still", "headset", "right", "try", "color", "small", "contact"	experiences with mobile phones and accessories regarding design, functionality, and overall satisfaction. Customers discuss screen quality, aesthetics, and compactness; they show love for the products and disappointment with issues like fragility, poor reception, or not meeting expectations. **Quality:** This topic holds **10.9%** of total tokens (words) in the dataset with the most frequent term as 'I'.
4	"I", "use", "phone", "nice", "thing", "end", "two", "call", "headset", "cool", "could", "feature", "look", "sound", "wife", "cell", "day", "verizon", "love", "helpful", "disappointment", "come", "really", "hear", "new", "less", "leather", "simple", "case", "like"	**Interpretation:** This topic involves customer reviews related to mobile phones, headsets, and their accessories, such as leather cases. The reviews relate to features, design, and functionality, with customers narrating their experiences in making calls, sound quality, and aesthetic appeal. Quite often, customers comment on positive aspects like simplicity and useful features but express disappointment at times. **Quality:** This topic holds **10.9%** of total tokens (words) in the dataset with the most frequent term as 'I'.
5	"good", "quality", "ear", "sound", "fit", "poor", "price", "case", "volume", "say", "low", "audio", "even", "well", "real", "plug", "headset", "look", "nothing", "phone", "quite", "size", "make", "comfortable", "picture", "fine", "need", "loud", "use", "would"	**Interpretation:** This topic is about customer feedback related to headsets and audio devices, focusing mainly on the sound quality, fit, and price. Customers speak of good sound and comfortable fit, but they also mention poor audio quality, problems with volume, and disappointment.

		Quality: This topic holds **10.5%** of total tokens (words) in the dataset with the most frequent term as ‘good’.
6	"great", "work", "phone", "item", "charger", "product", "I", "take", "disappointed", "picture", "price", "anyone", "plug", "awesome", "nice", "time", "still", "also", "company", "sure", "device", "even", "look", "black", "really", "drop", "screen", "car", "impress", "part"	**Interpretation:** This topic covers customer feedback in regard to mobile phone-related products and accessories, like chargers, screens, and devices. The positive reviews speak highly of excellent performance, impressive features, and nice designs. However, other customers were quite disappointed with either the durability or functionality. **Quality:** This topic holds **9.5%** of total tokens (words) in the dataset with the most frequent term as ‘great’.
7	“I”, "recommend", "excellent", "would", "money", "waste", "headset", "return", "phone", "product", "highly", "people", "give", "price", "one", "voice", "back", "talk", "start", "send", "earpiece", "put", "right", "several", "hear", "even", "find", "problem", "unit", "try"	**Interpretation:** This topic covers the reviews of customers about headsets or earpieces regarding performance, value for money, and usability. Customers share their positive experiences in terms of good voice quality and satisfactory products, often recommending it to others. Negative feedback focuses on poor quality, problems at the time of use, or dissatisfaction leading to returns. Discussion also covers pricing, clarity of communication, and product fit or design. **Quality:** This topic holds **8.2%** of total tokens (words) in the dataset with the most frequent term as ‘I’.
8	"well", "service", "bad", "phone", "work", "charge",	**Interpretation:** This topic is related to reviews by customers

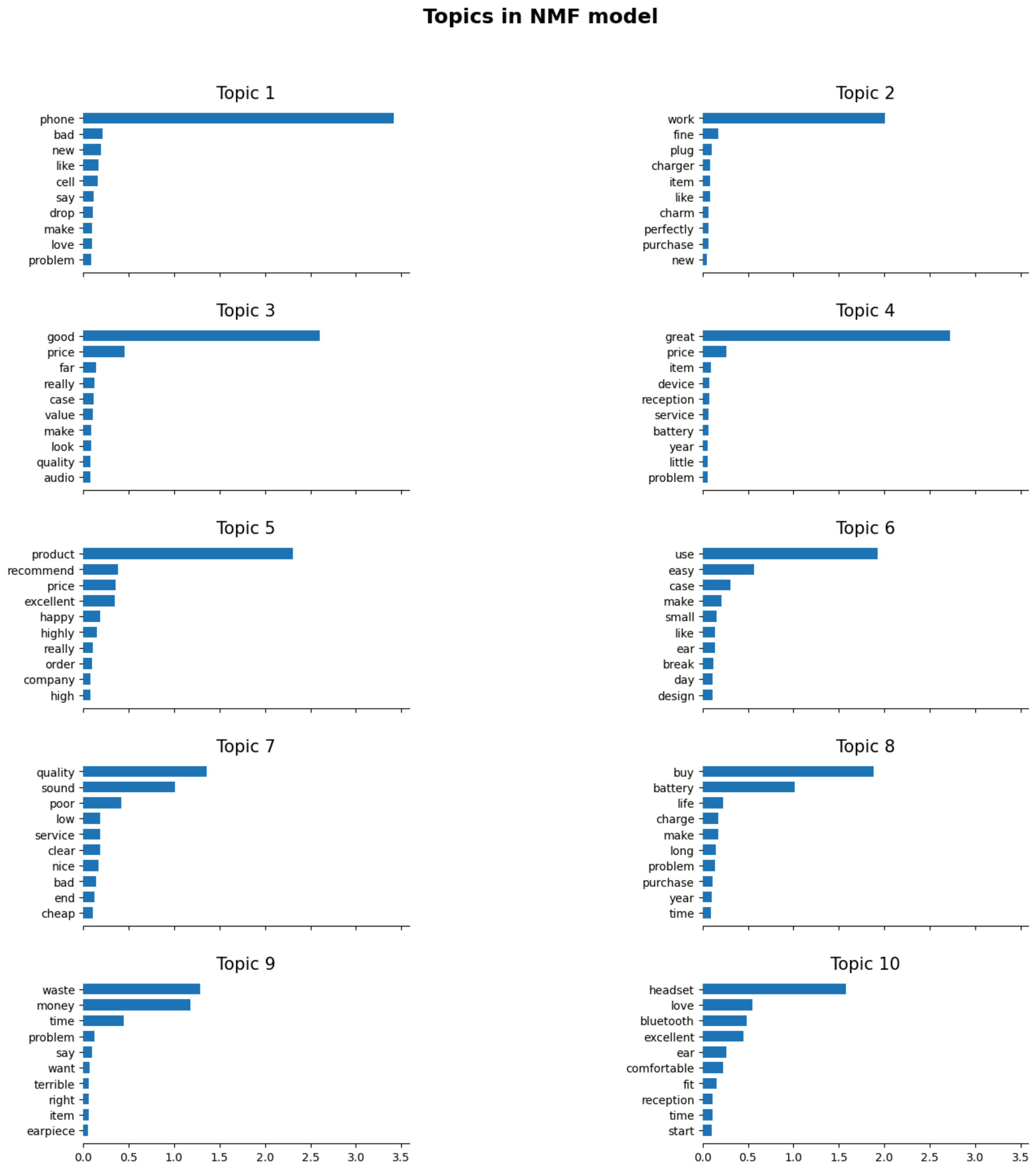
	"quality", "customer", "hold", "year", "comfortable", "sound", "keep", "clear", "ever", "camera", "clip", "crap", "wear", "belt", "turn", "I", "definitely", "time", "mobile", "terrible", "pretty", "first", "nokia", "completely"	about mobile phones, including Nokia, and their accessories concerning the service quality, performance, and durability. Some customers share positive feelings about sound clarity, comfort, and ease of use; other customers are frustrated with poor customer service, product quality, and the durability of accessories. Discussions often concern comments on battery performance, practical wearability-for instance, belt clips-and, finally, the general lifespan of devices and their accessories. **Quality:** This topic holds **8.1%** of total tokens (words) in the dataset with the most frequent term as 'well'.
9	"work", "phone", "I", "one", "case", "product", "like", "cheap", "lot", "feel", "light", "also", "software", "horrible", "side", "button", "charm", "drain", "new", "small", "none", "plastic", "would", "thing", "much", "little", "quickly", "player", "hand", "right"	**Interpretation:** This topic revolves around customer feedback on phones and accessories, with regard to build quality, design, and performance. Customers discuss physical attributes such as lightweight designs, compact sizes, and plastic materials, often comparing them to perceived product quality. Reviews highlight issues with software functionality, battery drain, and responsiveness of buttons, besides mentions of affordability and value. Positive comments praise the quick performance and comfortable handling, while negative comments criticize the cheap build and poor durability. **Quality:** This topic holds **8%** of total tokens (words) in the

		dataset with the most frequent term as ‘work’.
10	"battery", "easy", "time", "life", "make", "use", "long", "go", "last", "pair", "phone", "bad", "come", "tool", "know", "original", "clear", "excellent", "charge", "call", "couple", "earned", "stay", "button", "place", "still", "important", "enough", "especially", "also"	**Interpretation:** This topic is for reviews by customers about phone batteries or battery-related tools, considering aspects like longevity, ease of use, and quality. The positive ones described how the batteries lasted long, were easy to use, and their calls were clear. On the negative side, some customers complained about the performance, durability, or charging of the battery. Discussions included things like the dependability of a battery for everyday tasks and the value and quality of the product. **Quality:** This topic holds **7.5%** of total tokens (words) in the dataset with the most frequent term as ‘battery’.

4.2 Topic Detection using Non-Negative Matrix Factorization

- The tokenized texts are converted into a list.
- The texts are converted into TF-IDF representation.
- A non-negative matrix factorization model is trained to retrieve 10 topics.
- The topics are visualized.

Figure 25: ***Topics detected using NMF (Non-negative Matrix Factorization)***



Interpretation of the Detected Topics

Table 43: *Topic wise interpretation of the detected topics*

Topic No.	**Words Present**	**Interpretation of Topic and Quality Assessment**
1	"phone", "bad", "new", "like", "cell", "say", "drop", "make", "love", "problem"	**Interpretation:** This topic holds discussion around mobile phones which includes user dissatisfaction like “bad” and some positive sentiments like “love”. **Quality:** The most frequently occurring term is ‘phone’ and the least occurring term is ‘problem’
2	"work", "fine", "plug", "charger", "item", "like", "charm", "perfectly", "purchase", "new"	**Interpretation:** This topic focuses mainly on the functionality and performance of accessories like chargers, with mentions of satisfaction (working fine and perfectly). **Quality:** The most frequently occurring term is ‘work’ and the least occurring term is ‘new’
3	"good", "price", "far", "really", "case", "value", "make", "look", "quality", "audio"	**Interpretation:** This topic portrays focus on the price and value of items along with mentions of quality and audio-related products. **Quality:** The most frequently occurring term is ‘good’ and the least occurring term is ‘audio’
4	"great", "price", "item", "device", "reception", "service", "battery", "year", "little", "problem"	**Interpretation:** This topic is about reviews related to service, reliability, reception quality, and battery life. **Quality:** The most frequently occurring term is ‘great’ and the

		least occurring term is ‘problem’
5	"product", "recommend", "price", "excellent", "happy", "highly", "really", "order", "company", "high"	**Interpretation:** This topic holds positive product reviews and strong recommendations mostly highlighting happiness and high quality. **Quality:** The most frequently occurring term is ‘product’ and the least occurring term is ‘high’
6	"use", "easy", "case", "make", "small", "like", "ear", "break", "day", "design"	**Interpretation:** This topic mainly focuses on user experience, ease of use, compact design and durability of products. **Quality:** The most frequently occurring term is ‘use’ and the least occurring term is ‘dessign’
7	"quality", "sound", "poor", "low", "service", "clear", "nice", "bad", "end", "cheap"	**Interpretation:** This topic holds conversations around sound quality including mentions of poor or low performance and some positive feedback. **Quality:** The most frequently occurring term is ‘quality’ and the least occurring term is ‘cheap’
8	"buy", "battery", "life", "charge", "make", "long", "problem", "purchase", "year", "time"	**Interpretation:** This topic is related to customers’ concerns and opinions about battery life, charging efficiency and overall longevity of products. **Quality:** The most frequently occurring term is ‘buy’ and the least occurring term is ‘time’

9	"waste", "money", "time", "problem", "say", "want", "terrible", "right", "item", "earpiece"	**Interpretation:** This topic is about customers's complaints about wasted money and time along with dissatisfaction with items like earpieces. **Quality:** The most frequently occurring term is 'waste' and the least occurring term is 'earpiece'
10	"headset", "love", "Bluetooth", "excellent", "ear", "comfortable", "fit", "reception", "time", "start"	**Interpretation:** This topic holds positive feedback on Bluetooth headsets particularly regarding comfort, fit and reception quality. **Quality:** The most frequently occurring term is 'headset' and the least occurring term is 'start'

References

1. Explosion AI. (n.d.). *ContextualSpellCheck: A spell-checker using context and spaCy pipeline*. Retrieved January 10, 2025, from https://spacy.io/universe/project/contextualSpellCheck
2. Scikit-learn. (n.d.). *Scikit-learn: Machine learning in Python*. Retrieved January 10, 2025, from https://scikit-learn.org/stable/
3. Hugging Face. (n.d.). *Hugging Face documentation*. Retrieved January 10, 2025, from https://huggingface.co/docs
4. Hugging Face. (n.d.). *RoBERTa model documentation*. Retrieved January 10, 2025, from https://huggingface.co/docs/transformers/en/model_doc/roberta
5. Scikit-learn. (n.d.). *Topic extraction with non-negative matrix factorization and latent Dirichlet allocation*. Retrieved January 10, 2025, from https://scikit-learn.org/1.5/auto_examples/applications/plot_topics_extraction_with_nmf_lda.html