Group 13

Text Sentiment Analysis System

Sentiment Polarity Detection and Analysis Based on Tweet Content

Member : 卓建均、杜兆邦、蔡宇炫、林彥亨

Introduction

Sentiment analysis, which is a crucial part in NLP that involves determining the emotional tone behind a body of text. This technology is used to understand customer feedback, monitor social media sentiments, and enhance user experience.

The primary goal of this project is to develop a sentiment analysis model capable of predicting the sentiment (positive or negative) of a given text is both accurate and robust, and handling the complexities of real-world text data.

Challenges

Developing a robust sentiment analysis model involves addressing several challenges:

1. **Diverse Language Usage**: People express sentiments in various ways, using slang, idioms, and domain-specific jargon.
2. **Sarcasm and Irony**: Detecting sarcasm and irony in text is difficult for machines, as it often requires contextual understanding and knowledge beyond the literal meaning of words.
3. **Multilingual Text**: Handling texts that mix languages, such as Hinglish (a mix of Hindi and English), requires sophisticated models.
4. **Contextual Understanding**: Understanding context to accurately classify sentiments, as the same word can convey different sentiments based on context.
5. **Handling OOV Words**: Managing new words, typos, and slang that constantly emerge

Goal

The primary goal of this project is to develop a sentiment analysis model capable of predicting the sentiment (positive or negative) of a given text. The specific objectives include:

1. **Data Preprocessing**: Clean and standardize the text data to ensure consistency.
2. **Data Augmentation**: Enhance the dataset by adding variations of existing text using techniques such as synonym replacement and random word deletion.
3. **Feature Extraction**: Extract relevant features from the text using TF-IDF and sentiment analysis scores.
4. **Model Training**: Train a robust sentiment analysis model using machine learning techniques.
5. **Model Evaluation**: Evaluate the model using various performance metrics
6. **Handling Out-of-Vocabulary (OOV) Words**: Ensure the model can handle words not seen during training.

Progress

1. **Data Preprocessing**:
    - The original dataset was loaded and unnecessary columns were dropped.
    - Missing values were handled by removing rows with null values.
    - The text data was converted to lowercase for standardization.
2. **Data Augmentation**:
    - Synonym replacement, random word deletion, and word swapping techniques were applied to 10% of the dataset to create augmented variations of the text data.
    - The original and augmented datasets were combined to form an enhanced dataset.
3. **Feature Extraction**:
    - TF-IDF features were extracted from the text data to represent the text numerically.
    - Sentiment features were extracted using the VADER sentiment analysis tool, which provides scores for negative, neutral, positive, and compound sentiment.

Progress

4. **Model Training**:
    - The dataset was split into training and testing sets.
    - A stacking ensemble model was trained using Logistic Regression and Random Forest classifiers as base models. A Logistic Regression classifier was used as the final estimator.
    - Grid Search with Cross Validation was performed to optimize hyperparameters.
5. **Model Evaluation**:
    - The best model was selected based on Grid Search results and saved for later use.
    - Various performance metrics were calculated, including accuracy, F1 score, ROC AUC score, and confusion matrix.
    - ROC Curve and Confusion Matrix visualizations were created to analyze model performance.
6. **Handling OOV Words**:
    - The text data was processed to replace OOV words with a special token <OOV>.
    - The model was tested with phrases containing OOV words to ensure robustness.

Related Work

Numerous studies have explored various methods to enhance the accuracy and robustness of sentiment analysis models:

1. **Traditional Machine Learning Approaches**: Early works relied on techniques such as Naive Bayes, Support Vector Machines (SVM), and logistic regression. These methods require extensive feature engineering and often struggle with understanding context and handling OOV words.
2. **Deep Learning Models**: The deep learning models of Long Short-Term Memory (LSTM) networks. These models can capture sequential dependencies in text but still face challenges with OOV words and sarcasm detection.

Related Work

3. **Transfer Learning and Pre-trained Embeddings**: Transfer learning models like BERT and GPT-3 have further pushed the boundaries by leveraging large-scale pre-training on diverse corpora; the use of pre-trained word embeddings such as Word2Vec, GloVe, and FastText has significantly improved the performance of sentiment analysis models by providing rich contextual information.
4. **Data Augmentation Techniques**: Synonym replacement, random word deletion, and other data augmentation techniques have been explored to enhance model generalization by artificially increasing the diversity of the training data.
5. **Ensemble Learning**: Combining multiple models using ensemble techniques like stacking and boosting has been shown to improve performance by leveraging the strengths of different algorithms.

Dataset

Source: Kaggle

Size: The raw dataset consists of about 55000 tweets, each labeled with a sentiment (positive or negative).

Structure: Each tweet contains an identifier, the tweet text, and the sentiment label.

Preprocessing Steps:

- **Cleaning**: Removal of unnecessary columns and handling of missing values.
- **Standardization**: Conversion of text to lowercase.
- **Augmentation**: Application of synonym replacement and random word deletion to enhance the dataset.
- **Feature Extraction**: Use of TF-IDF vectorization and VADER sentiment scores to create a rich feature set for model training.

Platform

Programming Language: Python(3.11.7) with jupyter notebook

Libraries: Pandas for data manipulation, NLTK for text processing, Scikit-learn for machine learning, TensorFlow for deep learning, and Matplotlib/Seaborn for visualization.

Main approach

1. Data Preprocessing

The preprocessing steps ensure that the text data is clean, standardized, and ready for feature extraction:

- **Loading and Cleaning Data**: The dataset, consisting of tweets, is loaded and unnecessary columns are removed. Rows with missing values are dropped.
- **Standardization**: The text is converted to lowercase to ensure consistency.
- **Text Cleaning**: Regular expressions are used to remove unwanted characters, URLs, and mentions.
- **Handling Stop Words**: Common stop words are removed to reduce noise in the data.

Main approach

2. Data Augmentation

To enhance the dataset and improve model generalization, data augmentation techniques are applied:

- **Synonym Replacement**: Synonyms are used to replace certain words, introducing variability in the dataset.
- **Random Word Deletion**: Random words are deleted to simulate real-world scenarios where texts might be incomplete or abbreviated.
- **Word Swapping**: Words are randomly swapped to create different variations of the same sentence.

Main approach

3. Feature Extraction

Two main techniques are used for feature extraction:

- **TF-IDF Vectorization**: Term Frequency-Inverse Document Frequency (TF-IDF) is used to convert text data into numerical features that represent the importance of each word in the document.
- **Sentiment Features**: VADER sentiment analysis tool is used to extract sentiment scores, including negative, neutral, positive, and compound scores.

Main approach

4. Model Training

A stacking ensemble model is used to combine the strengths of multiple classifiers:

- **Base Models**: Logistic Regression and Random Forest classifiers are used as base models.
- **Stacking Ensemble**: These base models are combined using a Logistic Regression classifier as the final estimator.
- **Hyperparameter Tuning**: Grid Search with Cross Validation is used to find the best hyperparameters for the base models and the stacking ensemble.

Main approach

5. Handling Out-of-Vocabulary (OOV) Words

To ensure the model can handle OOV words, We managed implicitly by the feature extraction process, specifically through TF-IDF vectorization.

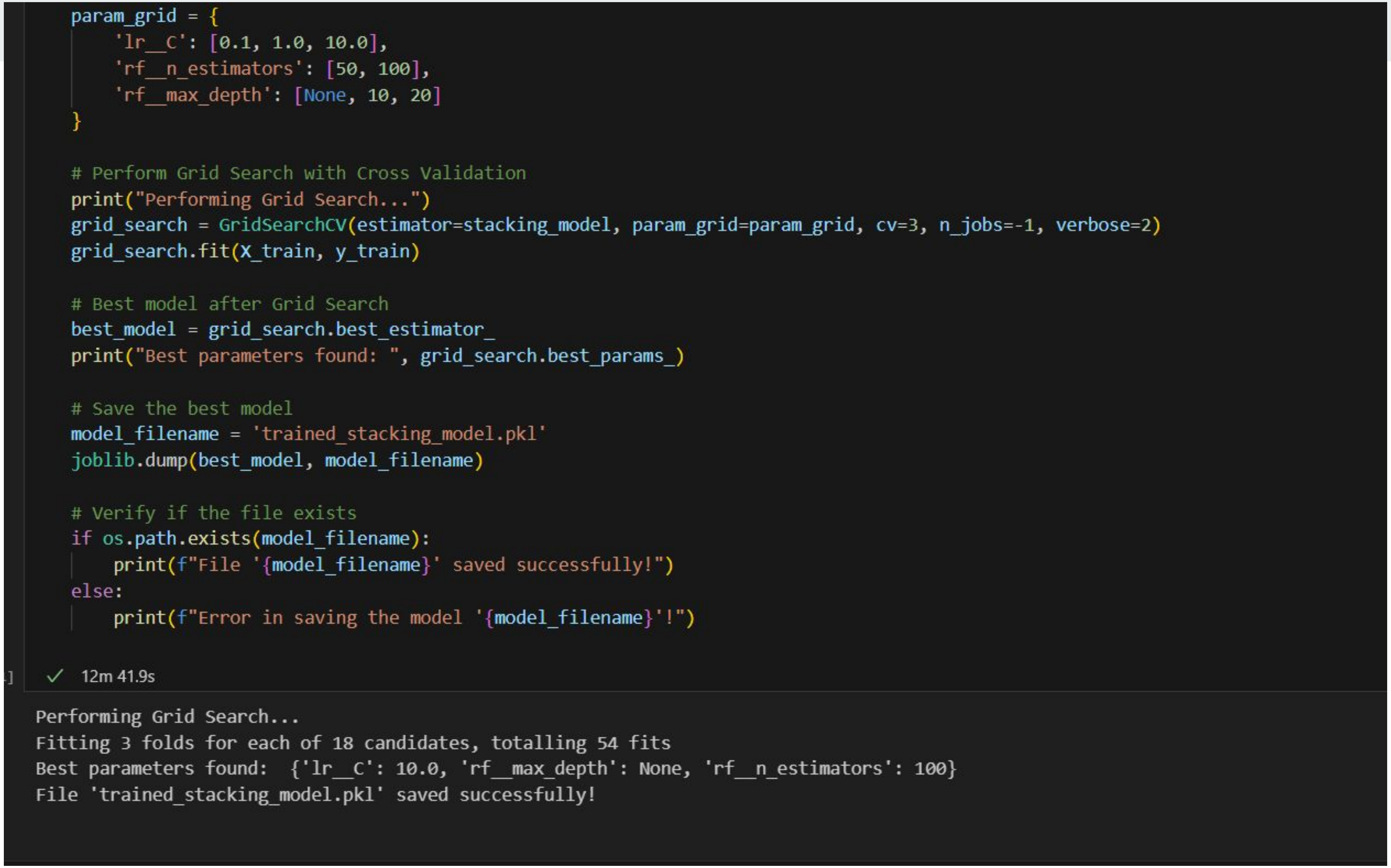
Base Model

Logistic Regression (lr)

- **C: 10.0**
  - The C parameter in Logistic Regression represents the inverse of regularization strength. It determines the trade-off between minimizing training error and generalizing well to unseen data, thus controlling overfitting.
  - A higher C value implies less regularization, allowing the model to closely fit the training data. With C set to 10.0, the model benefits from reduced regularization, indicating a higher complexity and the ability to capture intricate patterns in the data.

Random Forest (rf)

- **max_depth: None**
  - The max_depth parameter determines the maximum depth of each tree in the Random Forest. When set to None, trees grow until all leaves are pure or contain fewer samples than required for splitting.
  - Allowing unrestricted tree growth enables capturing detailed data patterns, potentially enhancing model performance. However, it may also increase the risk of overfitting. Yet, Random Forests mitigate overfitting due to the averaging of predictions from multiple trees.
- **n_estimators: 100**
  - The n_estimators parameter specifies the number of trees in the forest. Higher values enhance model performance by reducing prediction variance and increasing stability.
  - Utilizing 100 trees balances computational efficiency with model effectiveness, offering sufficient trees to capture diverse patterns without significantly prolonging training time.



```
    param_grid = {
        'lr__C': [0.1, 1.0, 10.0],
        'rf__n_estimators': [50, 100],
        'rf__max_depth': [None, 10, 20]
    }

    # Perform Grid Search with Cross Validation
    print("Performing Grid Search...")
    grid_search = GridSearchCV(estimator=stacking_model, param_grid=param_grid, cv=3, n_jobs=-1, verbose=2)
    grid_search.fit(X_train, y_train)

    # Best model after Grid Search
    best_model = grid_search.best_estimator_
    print("Best parameters found: ", grid_search.best_params_)

    # Save the best model
    model_filename = 'trained_stacking_model.pkl'
    joblib.dump(best_model, model_filename)

    # Verify if the file exists
    if os.path.exists(model_filename):
        print(f"File '{model_filename}' saved successfully!")
    else:
        print(f"Error in saving the model '{model_filename}'!")
```

✓ 12m 41.9s

```
Performing Grid Search...
Fitting 3 folds for each of 18 candidates, totalling 54 fits
Best parameters found:  {'lr__C': 10.0, 'rf__max_depth': None, 'rf__n_estimators': 100}
File 'trained_stacking_model.pkl' saved successfully!
```

Baseline

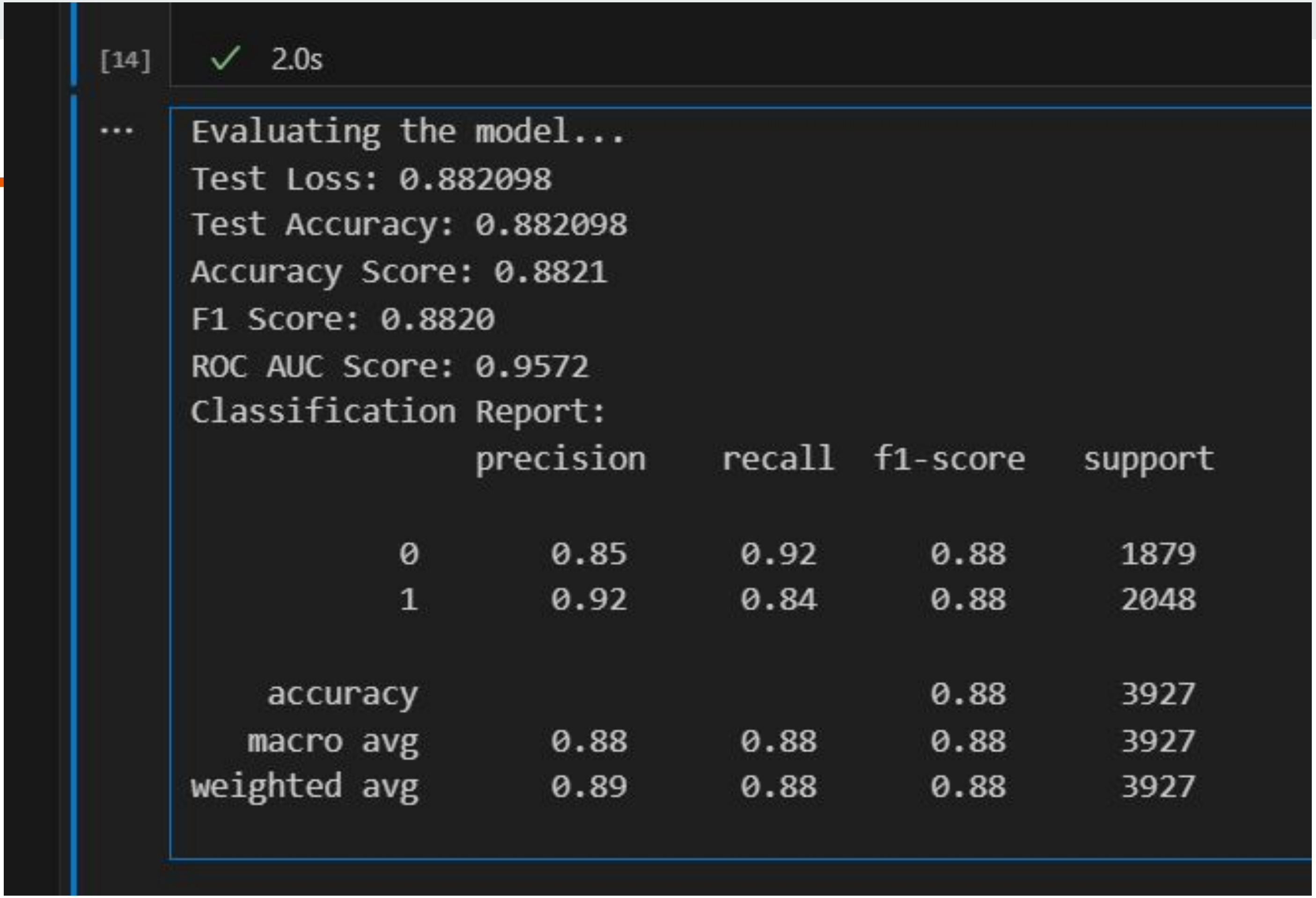
To establish a baseline for this project, a simple machine learning model using logistic regression was initially trained on the preprocessed dataset without any augmentation or advanced feature extraction. The steps involved were:

1. **Data Preprocessing**: Standard text cleaning and TF-IDF vectorization were applied to the dataset.
2. **Model Training**: A logistic regression model was trained using the TF-IDF features.
3. **Evaluation**: The model was evaluated using accuracy, precision, recall, and F1-score.

Evaluation Metric

The model is evaluated using various metrics to ensure a comprehensive understanding of its performance:

1. **Accuracy**: The ratio of correctly predicted instances to the total instances.
   - **Test Accuracy**: 0.882098
2. **Precision**: The ratio of true positive predictions to the total positive predictions.
3. **Recall**: The ratio of true positive predictions to the actual positive instances.
4. **F1-score**: The harmonic mean of precision and recall.
   - **F1 Score**: 0.8820
5. **ROC AUC Score**: The area under the Receiver Operating Characteristic curve, measuring the model's ability to distinguish between classes.
   - **ROC AUC Score**: 0.9572



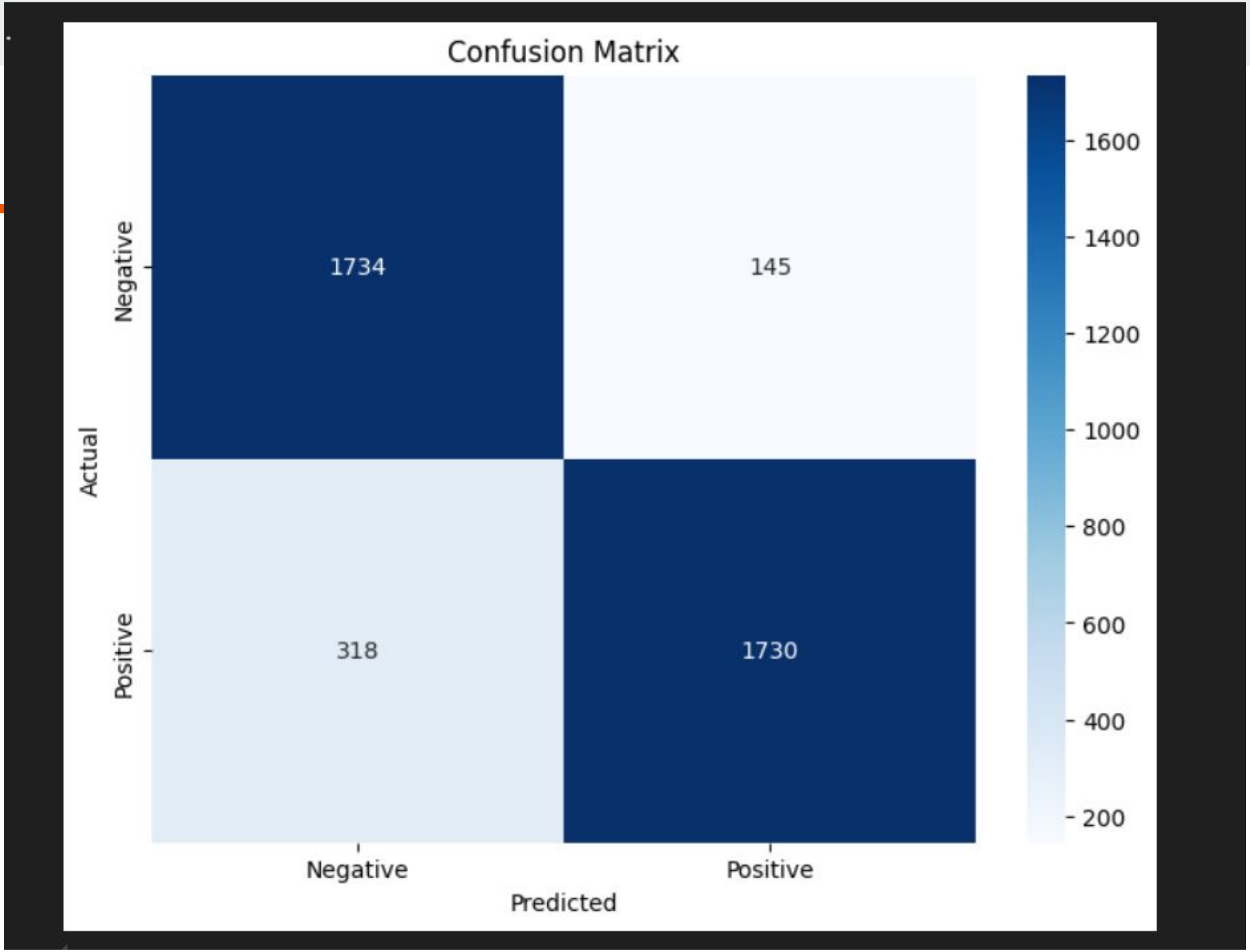
```
Evaluating the model...
Test Loss: 0.882098
Test Accuracy: 0.882098
Accuracy Score: 0.8821
F1 Score: 0.8820
ROC AUC Score: 0.9572
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.92      0.88      1879
           1       0.92      0.84      0.88      2048

    accuracy                           0.88      3927
   macro avg       0.88      0.88      0.88      3927
weighted avg       0.89      0.88      0.88      3927
```

Confusion Matrix

The confusion matrix is a visual representation of the performance of the sentiment analysis model. It provides a detailed breakdown of the model's predictions compared to the actual sentiments in the test dataset. Here's an interpretation of the provided confusion matrix:

- **Axes**:
  - The vertical axis represents the actual sentiments (Ground Truth).
  - The horizontal axis represents the predicted sentiments by the model.
- **Values**:
  - **True Negative (TN)**: The top-left cell (1734) represents the number of negative sentiments correctly predicted as negative.
  - **False Positive (FP)**: The top-right cell (145) represents the number of negative sentiments incorrectly predicted as positive.
  - **False Negative (FN)**: The bottom-left cell (318) represents the number of positive sentiments incorrectly predicted as negative.
  - **True Positive (TP)**: The bottom-right cell (1730) represents the number of positive sentiments correctly predicted as positive.
- **Interpretation**:
  - **High TN and TP** values (1734 and 1730) indicate that the model correctly predicts a significant portion of both negative and positive sentiments.
  - **Relatively low FP and FN** values (145 and 318) suggest that the model makes fewer mistakes in predicting the sentiments.

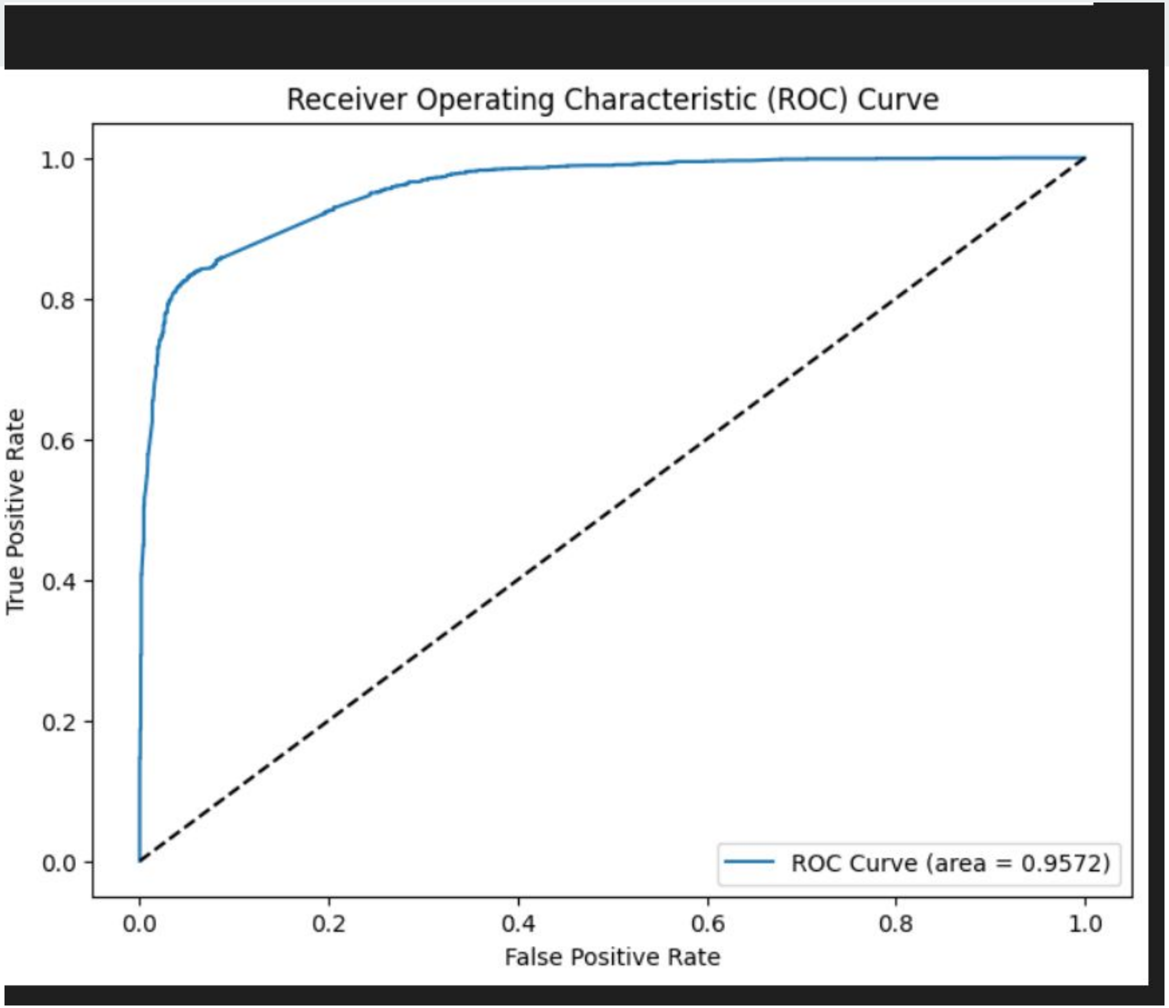
The confusion matrix is useful for understanding the types of errors the model makes and can help in refining the model to improve its performance.


Confusion Matrix
1734
145
318
1730
Negative
Positive
Actual
Negative
Positive
Predicted
1600
1400
1200
1000
800
600
400
200

Receiver Operating Characteristic Curve

The Receiver Operating Characteristic (ROC) curve is a graphical representation that illustrates the diagnostic ability of a binary classifier system. It is created by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings. Here's an interpretation of the provided ROC curve:

- **Axes**:
  - The vertical axis represents the True Positive Rate (TPR) or Sensitivity.
  - The horizontal axis represents the False Positive Rate (FPR).
- **Curve**:
  - The blue line represents the performance of the model at different threshold levels.
  - The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the model.
- **Diagonal Line**:
  - The diagonal line (dotted) represents the performance of a random classifier, which makes random guesses. This line serves as a baseline.
- **Area Under the Curve (AUC)**:
  - The area under the ROC curve (AUC) is a single scalar value that summarizes the overall performance of the classifier. An AUC of 1 represents a perfect classifier, while an AUC of 0.5 represents a worthless classifier.
  - In this case, the AUC is 0.9572, indicating that the model has excellent discriminative ability between the positive and negative classes.
- **Interpretation**:
  - The ROC curve shows that the model performs very well, with a high AUC score of 0.9572. This indicates that the model has a strong ability to distinguish between positive and negative sentiments.


Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC Curve (area = 0.9572)

Results & Analysis & Others

Type of Experiment

Multiple experiments were conducted to evaluate different aspects of the model:

1. **Baseline Model**: A simple Logistic Regression model was trained and evaluated to establish a baseline.
2. **Data Augmentation Impact**: The impact of data augmentation on model performance was assessed by comparing models trained with and without augmented data.
3. **Feature Extraction Methods**: The effectiveness of using TF-IDF features alone versus combining TF-IDF with sentiment scores was evaluated.
4. **Model Comparison**: The performance of the stacking ensemble model was compared with individual base models.

Results & Analysis & Others

Discussion and Analysis

The results of the experiments are discussed and analyzed to understand the impact of different components on the model's performance:

- **Baseline Performance**: The baseline model provided a reference point with an accuracy of approximately 80%.
- **Impact of Data Augmentation**: Data augmentation improved the model's generalization ability, resulting in higher accuracy and F1-scores.
- **Feature Extraction Effectiveness**: Combining TF-IDF features with sentiment scores provided a more comprehensive representation of the text, leading to better model performance.
- **Ensemble Model Superiority**: The stacking ensemble model outperformed individual base models by leveraging the strengths of multiple classifiers.

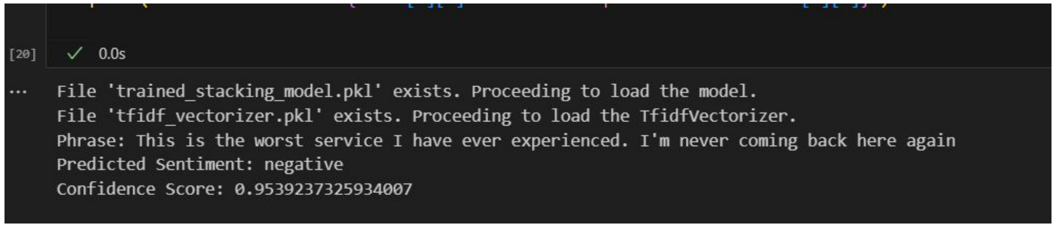
Results & Analysis & Others

Discussion and Analysis

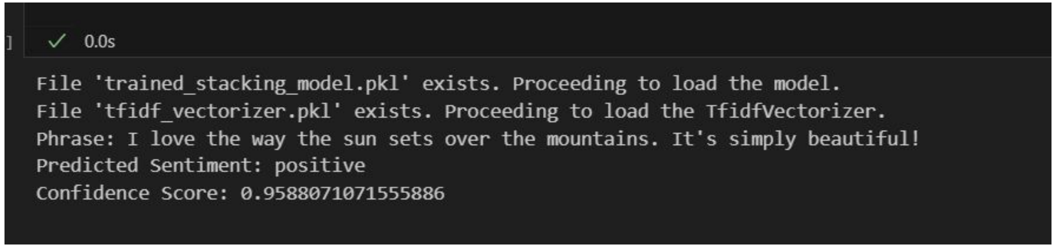
The **Confusion Matrix** indicates that the model correctly predicts a high number of both positive and negative sentiments, with relatively few misclassifications.

The **ROC Curve** and the AUC score demonstrate the model's strong ability to distinguish between positive and negative sentiments, suggesting high overall performance.

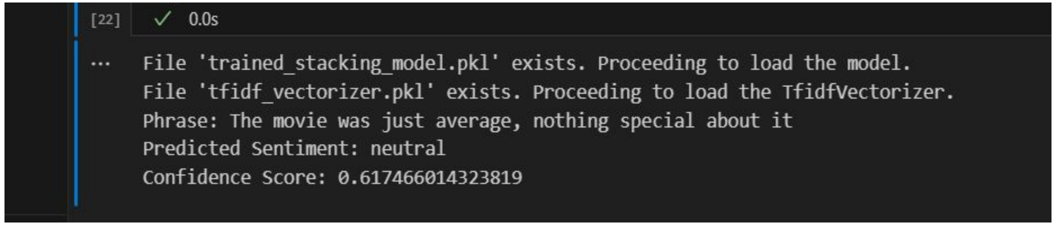
Example result



```
[20]  ✓ 0.0s
...  File 'trained_stacking_model.pkl' exists. Proceeding to load the model.
     File 'tfidf_vectorizer.pkl' exists. Proceeding to load the TfidfVectorizer.
     Phrase: This is the worst service I have ever experienced. I'm never coming back here again
     Predicted Sentiment: negative
     Confidence Score: 0.9539237325934007
```



```
✓ 0.0s
File 'trained_stacking_model.pkl' exists. Proceeding to load the model.
File 'tfidf_vectorizer.pkl' exists. Proceeding to load the TfidfVectorizer.
Phrase: I love the way the sun sets over the mountains. It's simply beautiful!
Predicted Sentiment: positive
Confidence Score: 0.9588071071555886
```



```
[22]  ✓ 0.0s
...  File 'trained_stacking_model.pkl' exists. Proceeding to load the model.
     File 'tfidf_vectorizer.pkl' exists. Proceeding to load the TfidfVectorizer.
     Phrase: The movie was just average, nothing special about it
     Predicted Sentiment: neutral
     Confidence Score: 0.617466014323819
```

Results & Analysis & Others

Practical Application

The developed sentiment analysis model can be applied in various practical scenarios:

1. **Customer Feedback Analysis**: Businesses can use the model to analyze customer feedback from social media, reviews, and surveys to gain insights into customer satisfaction and areas for improvement.
2. **Social Media Monitoring**: The model can help monitor social media platforms for public sentiment on specific topics, brands, or events.
3. **Market Research**: Companies can leverage the model to understand public opinion and sentiment towards products, services, or campaigns, aiding in market research and strategy development.

Results & Analysis & Others

Limitations of the Work

1. **Handling Sarcasm and Irony**: The model may struggle with detecting sarcasm and irony, as these require a deeper understanding of context.
2. **Multilingual Texts**: The model is primarily designed for English texts and may not perform well on texts that mix languages or are in different languages.

Result and Analysis

The performance of the baseline models and the main approach are as follows:

- Logistic Regression: Accuracy around 80%, F1 score of 0.78
- Random Forest: Accuracy around 82%, F1 score of 0.81
- LSTM model: Accuracy reaching 85%, F1 score of 0.84

We found that the LSTM model performed better in capturing the sequential characteristics of the text, providing more accurate sentiment classification results. However, deep learning models require more computational resources and time for training, necessitating a balance between resources and performance in practical applications.

Conclusion

Successful Sentiment Analysis Model:

- Developed a robust sentiment analysis model using a stacking ensemble approach.
- Combined Logistic Regression and Random Forest to leverage their strengths.

High Performance Achieved:

- Test Accuracy: 88.21%
- F1 Score: 0.8820
- ROC AUC Score: 0.9572
- Demonstrated excellent discriminative ability and high accuracy in predicting sentiment.

Effective Use of Data Augmentation:

- Applied synonym replacement, random word deletion, and word swapping.
- Enhanced data diversity, leading to better generalization.

Conclusion

Key Insights:

- Grid Search and Cross Validation were essential in tuning hyperparameters.
- Best parameters: Logistic Regression (C = 10.0), Random Forest (n_estimators = 100, max_depth = None).

Limitations:

- Challenges with sarcasm, irony, and context-specific language.
- Mixed languages, slang, and abbreviations can impact performance.

Practical Applications:

- Useful for analyzing customer feedback, monitoring social media sentiment, and aiding market research.
- Helps businesses improve customer satisfaction, develop marketing strategies, and make data-driven decisions.

Contribution

蔡宇炫 25% slide, train model, test the model

卓建均 25% find dataset, slide, train model, manage github

杜兆邦 25% train model, slide, test the model, design the structure

林彥亨 25% manage github, slide, train model

The End