Mini Project 2

ABO AKADEMI UNIVERSITY
Department of Informatics Technology

Machine Learning
Year 2023

Student Lamin Jatta

Contents

0.1 Intro

Sentiment analysis, also known as opinion mining, is a subfield of Natural Language Processing (NLP) that involves the computational treatment of opinions, sentiments, and emotions expressed in text. There are various approaches to sentiment analysis, and the techniques used can be classified into two broad categories: lexicon-based methods and machine learning-based methods.

- Lexicon-based methods involve mapping words in the text to their corresponding sentiment scores in a predefined lexicon or dictionary. This method is simple and fast, but it is limited by the quality and coverage of the sentiment lexicon used.
- Machine learning-based methods use supervised learning algorithms to train a model on a large annotated dataset and make predictions about the sentiment of the new, unseen text. This method is more flexible and can capture complex relationships between words and their sentiments, but it requires a large labelled dataset for training.

In addition to these two main categories, there are also hybrid methods that combine the strengths of both lexicon-based and machine learning-based approaches. Sentiment analysis can also be applied at different levels of granularity, such as document-level sentiment, sentence-level sentiment, and aspect-level sentiment analysis.
The sentiment of the following tweets ranges from positive (4) to neutral (0) to negative (0).
Data exploration is done in the script file. I did not add it here is not necessary.

0.2 Machine Learning Models

Machine learning models that can be used for sentiment analysis on the given dataset:

1. Naive Bayes Classifier: Naive Bayes is a popular and simple algorithm that can be used for sentiment analysis. It uses Bayes' theorem to calculate the probability of a tweet belonging to a particular sentiment class. In this case, the sentiment classes can be positive (4) and negative (0). The algorithm calculates the probabilities of words appearing in positive or negative sentiment classes and uses these probabilities to classify new tweets.
2. Support Vector Machines (SVM): SVM is another popular algorithm that can be used for sentiment analysis. SVM maps the input data into high-dimensional feature space and finds a hyperplane that separates the data into different classes. In this case, the classes can be positive (4) and negative (0). SVM maximizes the margin between the two classes, making it a robust classifier for sentiment analysis. The algorithm uses feature extraction techniques, such as TF-IDF, to represent the tweets as vectors of numbers.

Both algorithms can be implemented using popular machine learning libraries, such as scikit-learn in Python.
In this assignment, I tried different approaches to model the outcome of the sentiment 1 for YES and 0 for NO.

0.2.1 Data Preprocessing

Preprocess the raw text data by removing stop words, stemming or lemmatizing words, and converting the text into a numeric representation such as one-hot encoding or word embeddings.

0.2.2 Data Splitting

Split the data into a training set and a testing set, usually with a 70-30 or 80-20 split.

0.2.3 Training the Model

Train a logistic regression model using the training data. You can use a package such as scikit-learn to train the model. You may want to experiment with different hyperparameters such as the regularization parameter to optimize the performance of the model.

0.2.4 Model Evaluation

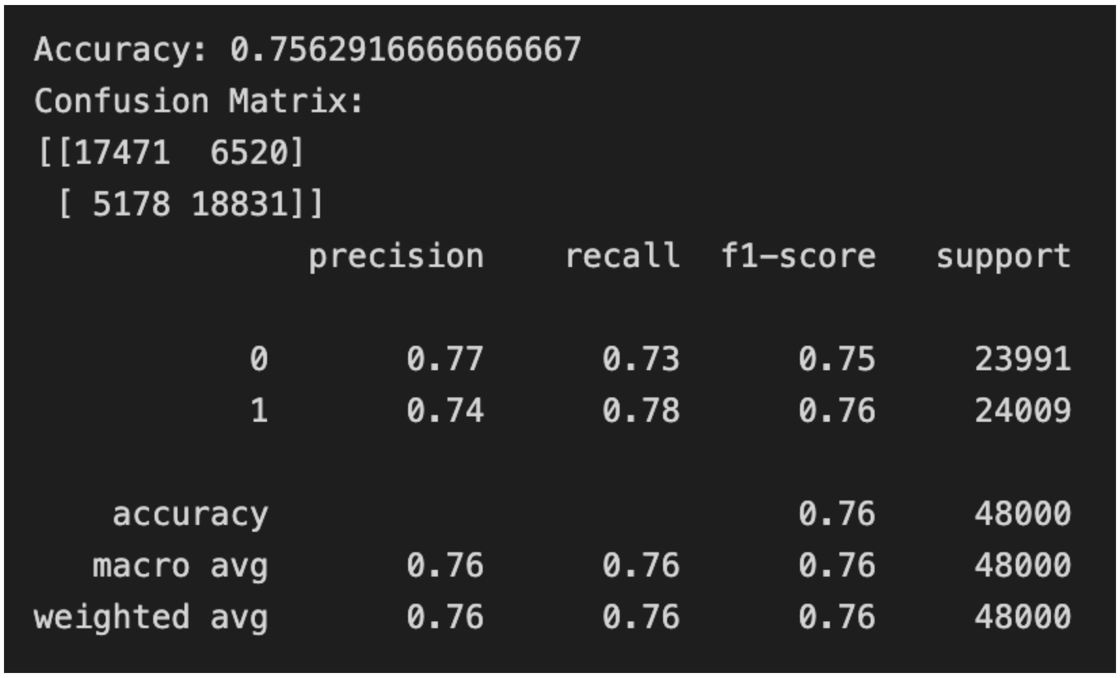
Evaluate the performance of the model using the testing data. Calculate metrics such as accuracy, precision, recall, and F1 score to determine how well the model is performing.

0.2.5 Predictions

Use the trained model to predict the sentiment of new, unseen text data.

0.3 Logistic regression model

Logistic Regression is a statistical model used for binary classification, which means it's used to predict a binary output (such as ”yes” or ”no”, ”positive” or ”negative”, ”spam” or ”not spam”) based on one or more input features. It uses a logistic function (also called a sigmoid function) to model the probability of the binary output given the input features.



```
Accuracy: 0.7562916666666667
Confusion Matrix:
[[17471  6520]
 [ 5178 18831]]
              precision    recall  f1-score   support

           0       0.77      0.73      0.75     23991
           1       0.74      0.78      0.76     24009

    accuracy                           0.76     48000
   macro avg       0.76      0.76      0.76     48000
weighted avg       0.76      0.76      0.76     48000
```

Figure 1: Logistic regression score.

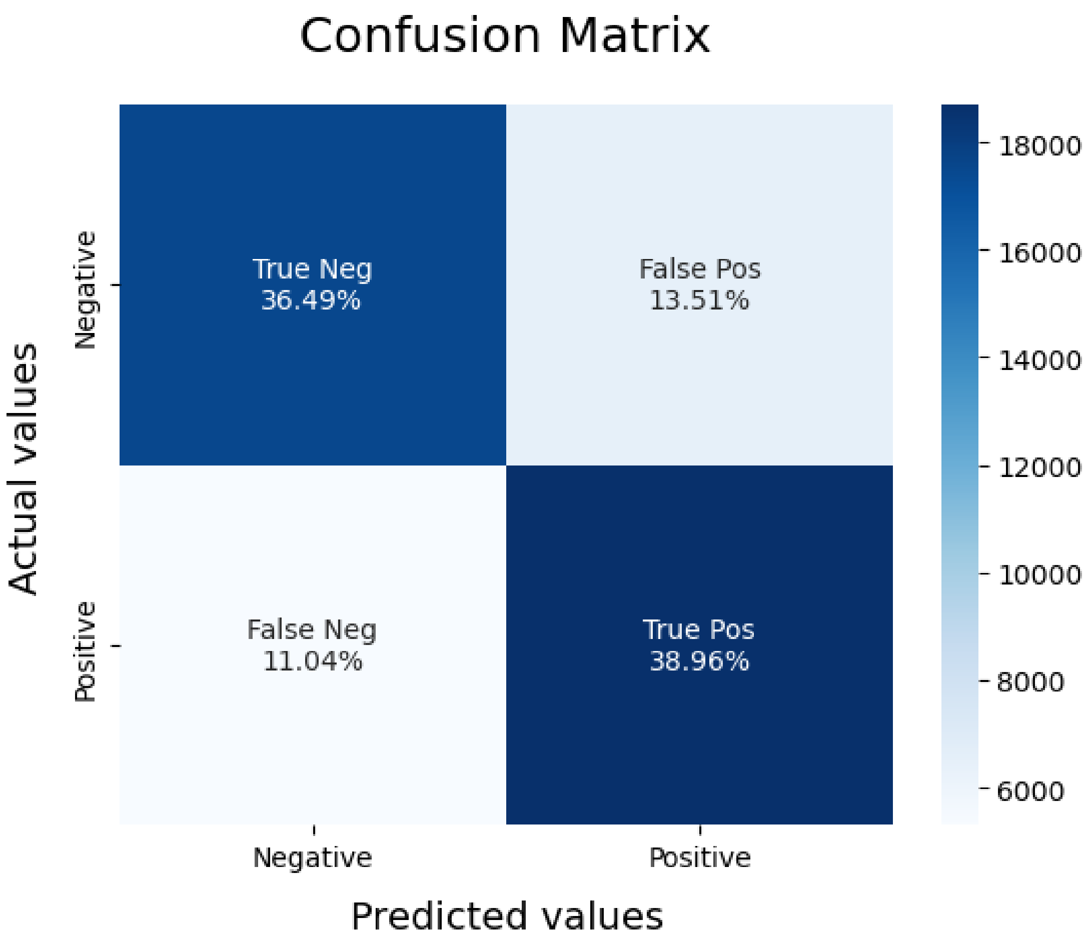
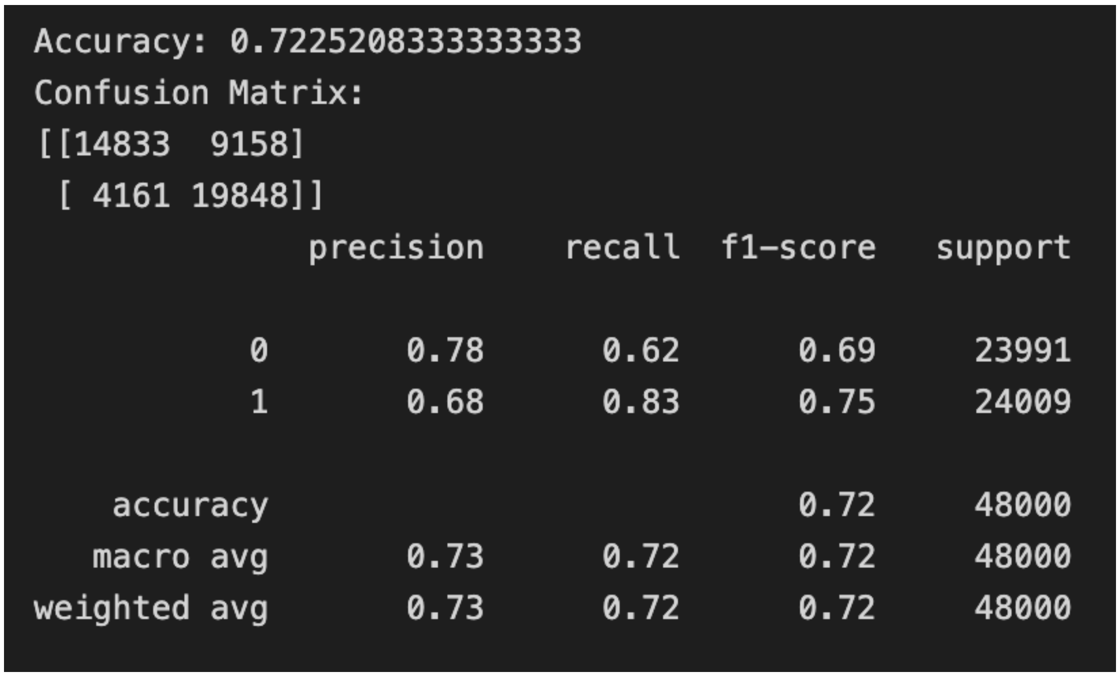


Figure 2: Logistic regression confusion matric.

0.4 XGBoost model

XGBoost for sentiment analysis in a similar way as logistic regression. XGBoost is a popular machine-learning library that is often used for classification and regression problems. It is an ensemble method that combines multiple decision trees to make predictions and is known for its high accuracy and fast performance.

We first preprocess the data and split it into training and testing sets, and then train an XGBoost model on the training set using scikit-learn's XGBClassifier. We then evaluate the performance of the model on the testing set by calculating the accuracy and creating a confusion matrix using scikit-learn's confusion_matrix function.



```
Accuracy: 0.7225208333333333
Confusion Matrix:
[[14833  9158]
 [ 4161 19848]]
              precision    recall  f1-score   support

           0       0.78      0.62      0.69     23991
           1       0.68      0.83      0.75     24009

    accuracy                           0.72     48000
   macro avg       0.73      0.72      0.72     48000
weighted avg       0.73      0.72      0.72     48000
```

Figure 3: XGBoost score.

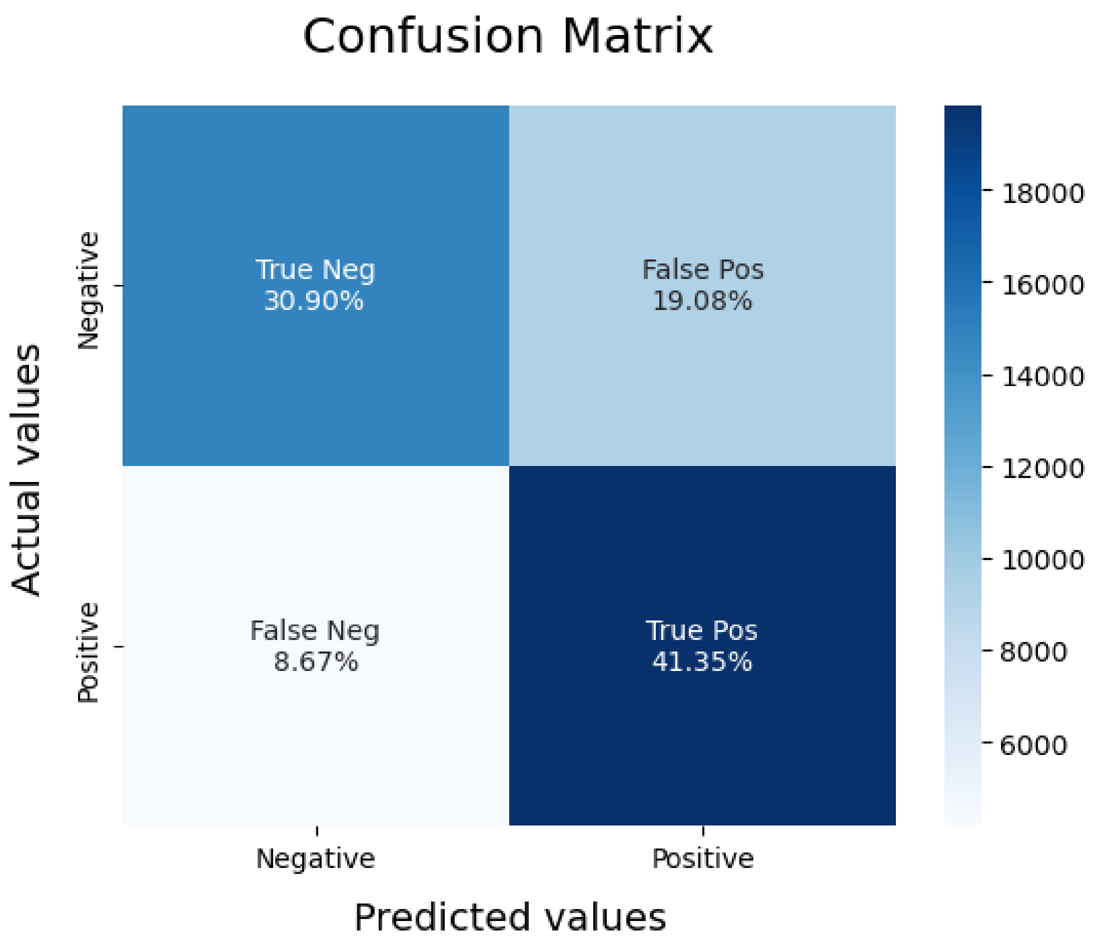
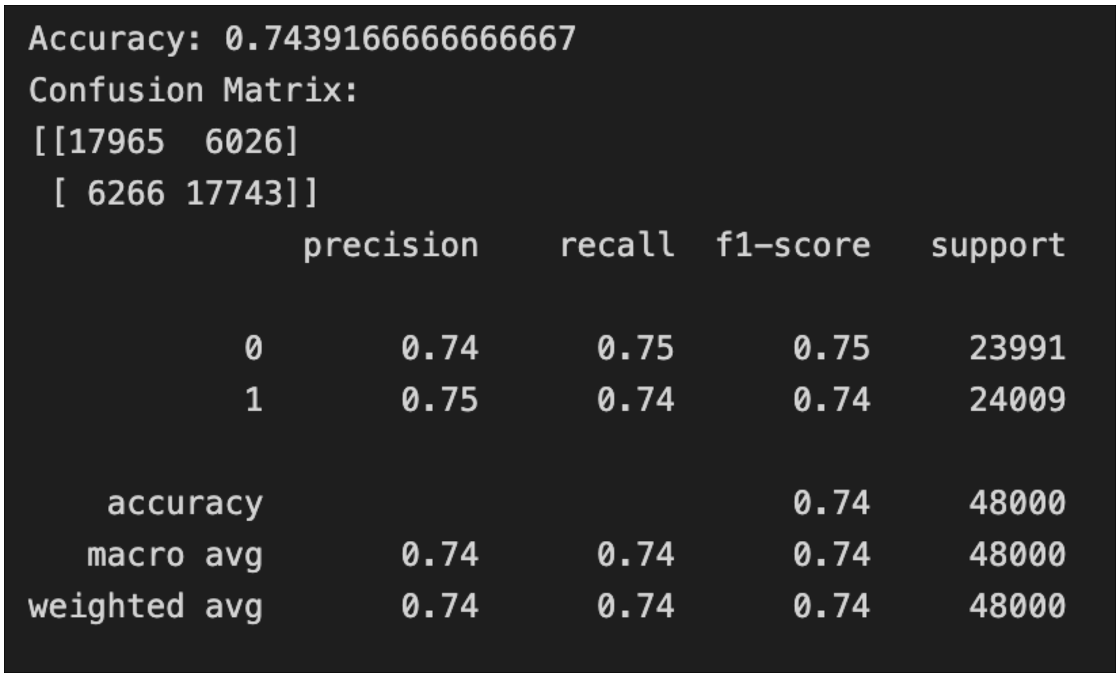


Figure 4: XGBoost confusion matric.

0.5 Random forest model

We first preprocess the data and split it into training and testing sets, and then train a Random Forest model on the training set using scikit-learn's Random-ForestClassifier. We then evaluate the performance of the model on the testing set by calculating the accuracy and creating a confusion matrix using scikit-learn's confusion_matrix function.



```
Accuracy: 0.7439166666666667
Confusion Matrix:
[[17965  6026]
 [ 6266 17743]]
              precision    recall  f1-score   support

           0       0.74      0.75      0.75     23991
           1       0.75      0.74      0.74     24009

    accuracy                           0.74     48000
   macro avg       0.74      0.74      0.74     48000
weighted avg       0.74      0.74      0.74     48000
```

Figure 5: Random forest score.

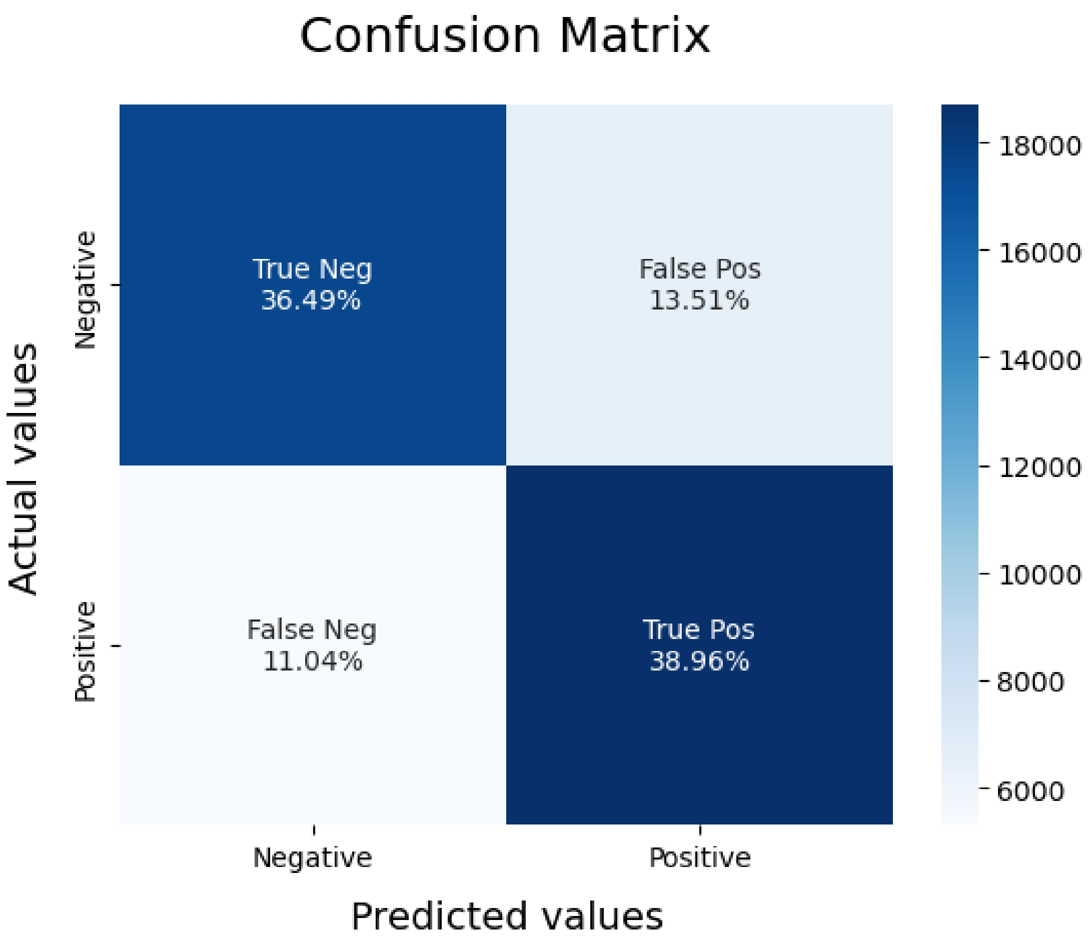
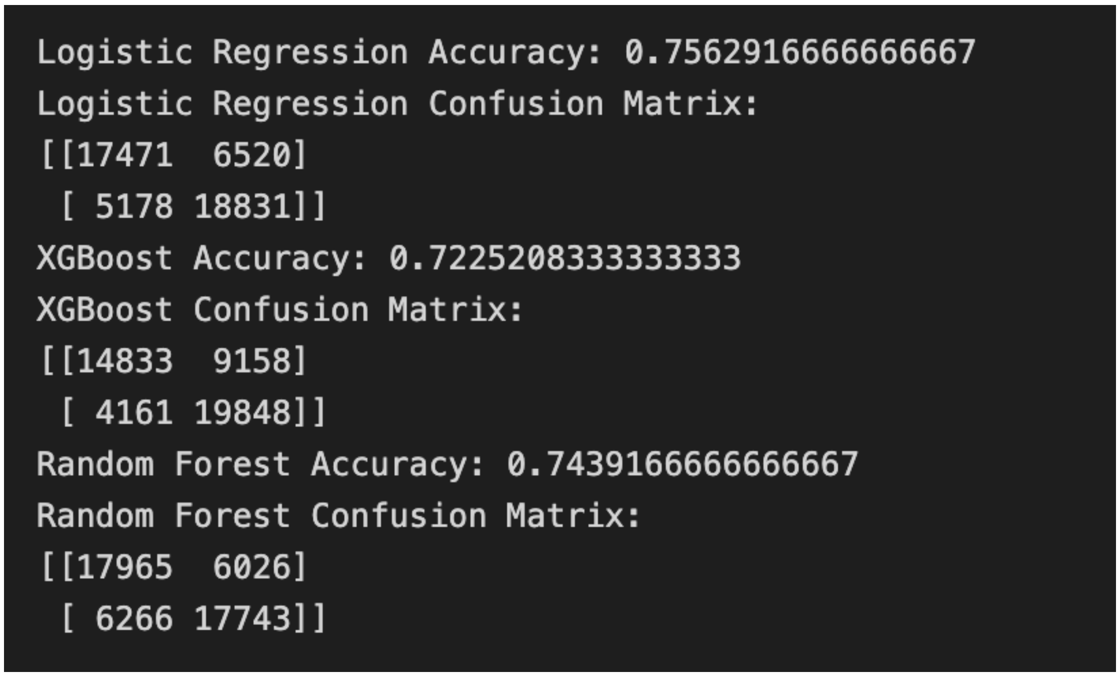


Figure 6: Random forest confusion matric.

0.6 Comparison



```
Logistic Regression Accuracy: 0.7562916666666667
Logistic Regression Confusion Matrix:
[[17471  6520]
 [ 5178 18831]]
XGBoost Accuracy: 0.7225208333333333
XGBoost Confusion Matrix:
[[14833  9158]
 [ 4161 19848]]
Random Forest Accuracy: 0.7439166666666667
Random Forest Confusion Matrix:
[[17965  6026]
 [ 6266 17743]]
```

Figure 7: Comparison of models.

From the accuracy and confusion matrix for three different models: Logistic Regression, XGBoost, and Random Forest.
The accuracy measures how well the models are able to correctly classify the input data. The accuracy scores for the three models are 0.756 for Logistic Regression, 0.723 for XGBoost, and 0.744 for Random Forest. These scores indicate that Logistic Regression performed the best among the three models in terms of accuracy.
The confusion matrix shows the number of correct and incorrect predictions made by each model. The confusion matrices for the three models are shown below:
Logistic Regression Confusion Matrix:

	F	T
F	17471	6520
T	5178	18831

XGBoost Confusion Matrix:

	F	T
F	14833	9158
T	4161	19848

Random Forest Confusion Matrix:

	F	T
F	17965	6026
T	6266	17743

In each confusion matrix, the rows represent the actual classes and the columns represent the predicted classes. The numbers in the matrix represent the number of instances that fall into each category. For example, in the Logistic Regression

confusion matrix, there were 17,471 instances that were correctly predicted as the negative class (the first row, first column), 6,520 instances that were incorrectly predicted as the positive class (the first row, second column), 5,178 instances that were incorrectly predicted as the negative class (the second row, first column), and 18,831 instances that were correctly predicted as the positive class (the second row, second column). The same interpretation applies to the other two confusion matrices.
Overall, based on the accuracy scores and confusion matrices, Logistic Regression appears to be the best model among the three for this particular task. However, it's always important to consider multiple metrics and do a more thorough analysis before making any final conclusions.

0.7 Other Useful models worth try

0.7.1 CNN model

Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) can both be used for sentiment analysis, but they have different strengths and weaknesses.
CNNs are particularly good at processing large amounts of data and handling fixed-length inputs, such as images or text with a fixed number of words. For sentiment analysis of text, a 1D convolutional layer can be used to extract features from the words and phrases in the input, followed by one or more fully connected layers to perform the classification.

0.7.2 RNN model

RNNs, on the other hand, are well suited for processing sequential data, such as time series or text with variable length. For sentiment analysis of text, an RNN can be trained to encode the meaning of the words in a sequence, taking into account the context of the surrounding words. The hidden state of the RNN can then be used as the feature representation for the text, and a fully connected layer can be used to perform the classification.
In terms of comparison, each model has its own advantages and disadvantages, and the choice of which one to use depends on the specific requirements of the task and the data. In general, CNNs tend to be faster and easier to train than RNNs, but they may not be able to capture long-range dependencies as well as RNNs. On the other hand, RNNs can be more computationally expensive and prone to overfitting, but they can handle longer sequences and capture more complex relationships between the words in the text.
In terms of which is better, it really depends on the task and the data. Both CNNs and RNNs have been used to achieve state-of-the-art results on various sentiment analysis benchmarks, so it's a matter of experimenting with both models and choosing the one that works best for your specific use case.

0.8 Conclusion

Based on the results of the analysis, it can be concluded that Logistic Regression is the best model for the sentiment analysis task in this particular scenario. This

conclusion is based on the model's high accuracy score of 0.76 and the relatively low number of incorrect predictions in the confusion matrix.
However, it's important to note that the other two models, XGBoost and Random Forest, also performed reasonably well, with accuracy scores of 0.72 and 0.74, respectively. Therefore, it may be worth considering these models in other contexts where different metrics or trade-offs are important.
Overall, the choice of the best model depends on various factors, including the dataset, the task, the available resources, and the desired performance metrics. Therefore, it's recommended to try different models and compare their performance on the specific task at hand before making a final decision.