Natural Language Proccessing

AIVC

MSc in

Artificial Intelligence &
Visual Computing

Rubati Mateo
AIVC24015

Project Description

Sentiment Analysis is a critical natural language processing task that focusses on extracting and understanding the sentiments associated with a product or service mentioned in text, such as product reviews. The aim of this project is to guide you through the process of designing, implementing, and evaluating sentiment analysis models using various NLP techniques, including traditional machine learning, deep learning, and transfer learning models. During the project you will have the opportunity to explore the entire pipeline from data collection, preprocessing, model training, and evaluation, to fine-tuning advanced models such as transformers.

Objectives:

- Gain practical experience in sentiment analysis for domain-specific texts.
- Understand the challenges of data collection, labelling, and preprocessing.
- Explore various modelling approaches (e.g., classical machine learning models, deep learning models like LSTMs or transformers).
- Evaluate model performance and interpret the results.

TABLE OF CONTENTS

Project Description 2
1. Domain Selection and Dataset Creation 5
2. Preprocessing and Text Analysis 7
3. Feature Extraction 8
4. Model Development and Evaluation 9
Baseline models 9
Baseline models evaluation 9
Deep Learning Models 13
Model Implementation 13
Deep learning models evaluation 15
5. Comparisons And Evaluation 19
Model Comparison 19
Preprocessing Methods Comparison 21
Diffrences Between Baseline and Deep Learning Models 22
6. Conclusions 23
7. Future Work 23
Appendix 24
1. Tables and metrics 24
2. Libraries and Tools 30
3. Links 30

1. Domain Selection and Dataset Creation

The domain chosen for sentiment analysis is the domain of finance and the classification of financial news into positive and negative news. The dataset used consists of 5322 sentences related to financial news and has sentiment labels related to positive, negative and neutral news. For the purpose of the paper the neutral class is not used in the analysis and the new dataset consists of 2712 news of which 1852 are in the positive class and 860 in the negative class.

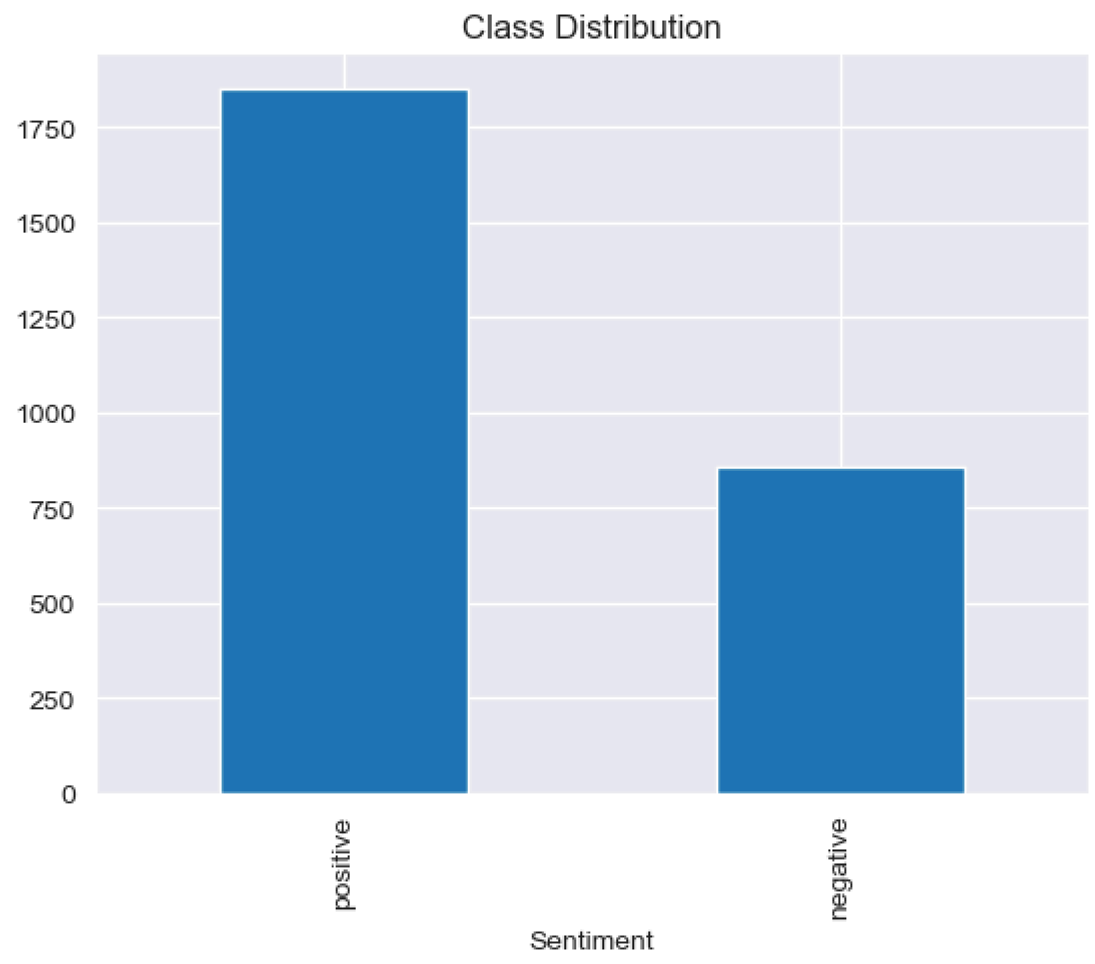

Image 1: Class distribution

Table 1: Text length statistics

Statistic	Value
Count	2712
Mean	109.99
Standard Deviation (std)	55.07
Minimum (min)	10
25th Percentile (25%)	67
Median (50%)	98
75th Percentile (75%)	140
Maximum (max)	298

2. Preprocessing and Text Analysis

Preprocessing is a critical step in any NLP task. It helps to clean the text data and makes it suitable for analysis. In this project the following steps have been taken in preprocessing :

- Remove extra whitespaces
- Remove html tags
- Remove URLS
- Decode Html entities
- Replace stock tickers (e.g., $AAPL) with a special token 'STOCK'
- Handle currencies (e.g., $10k, €20)
- Remove space between digits and decimal points
- Remove space before or after a comma in large numbers
- Ensure that numbers with units (e.g., EUR 1.6 m) have no spaces
- Replace percentages
- Replace months, dates, years, and hours
- Remove special characters
- Remove punctuation

There is also the use of stemming and lematazing as we want to examine how these two methods affect our results. We also examine how stopwords also affect the results of the experiment. To achieve what has been said above, the following tools were used:

1. Regular expressions for the representation of symbols and words
2. The nltk.stem library for using the porterstemmer and wordnetlemmatizer
3. The nltk.corpus library for stopwords
4. And the library nltk.tokenize for word_tokenize

3. Feature Extraction

The next step after preprocessing the data is the extraction of features. In this particular experiment the following methodology was followed:

1. First the data were split into training data and test data.
2. Then the training data was used to extract the features using tf idf vectors
3. After the feature extraction, the oversampling of the smallest class was done by using SMOTE
4. Finally the test data is converted into tf idf vector as well

Features are extracted only from the training data to avoid data leakage into the validation set, ensuring that the model is evaluated on unseen data. Once the features are extracted, the training and validation data are vectorized separately. The oversampling is done in order to solve the problem of class imbalance so that we create a more balanced dataset, improving the model's ability to learn meaningful patterns for both positive and negative sentiment classifications. So it does not lead to a model that is biased toward predicting the majority class sentiment. Also by oversampling after the extraction of the features we ensure that we will not have synthetic features to train the model.

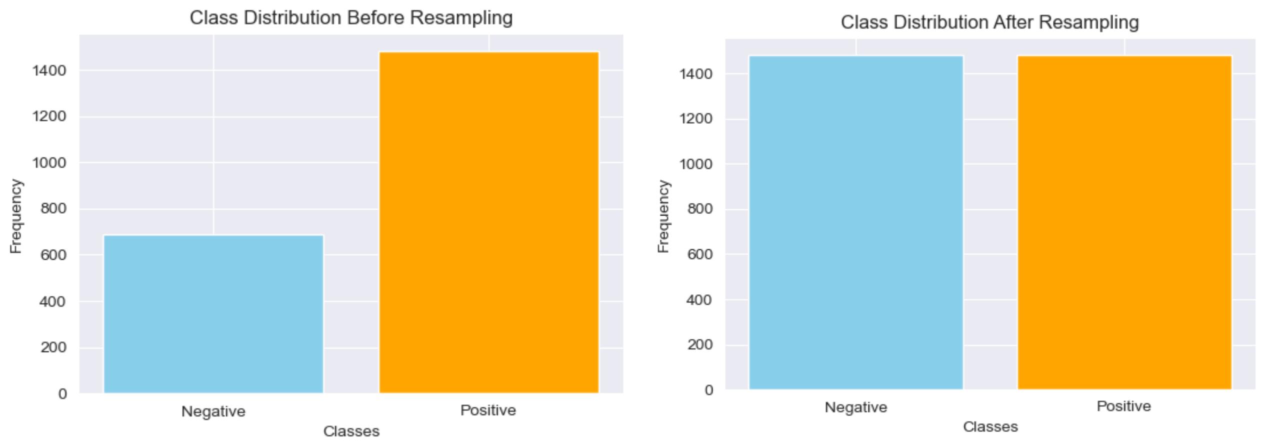

Image 2: Class distribution before and after oversampling

4. Model Development and Evaluation

The next and final step is the implementation of baseline models and deep learning models. The models used in this project are for baseline models Logistic Regression and SVM. And for the deep learning models are LSTM and CNN models.

Baseline models

The baseline models that are used as said earlier are the Logistic Regression model and the SVM model. They were picked due to the fact that they are popular for tasks like sentiment analysis and they are efficient and effective in text handling. Logistic Regression is fast, effective in binary classification and performs well with high dimensional data like TF-IDF and also has built-in regularization to prevent overfitting. On the other hand SVM is robust for small or imbalanced datasets, excels in handling high-dimensional and sparse data and also focuses on maximizing margins for better generalization and prevents the model from overfitting.

Baseline models evaluation

The models were trained and tested with 4 different preprocessing methods in order to explore the impact of different preprocessing techniques. This techniques are :

- Stemming without stopwords
- Stemming with stopwords
- Lemmatization without stopwords
- Lemmatization with stopwords

After the experiments[1] the results for just using stemming without removing the stopwords performed well compared to the other techniques. For the evaluation of the models metrics such us f1 score, accuracy and AUC were used.

[1] You can find more on that topic at part 5.

Classification Report Logistic Regression

	Precission	Recall	F1-Score	Support
Negative	0.76	0.74	0.75	172
Positive	0.88	0.89	0.89	371
Accuracy			0.85	543
Macro avg	0.82	0.82	0.82	543
Weighted avg	0.84	0.85	0.85	543

F1-Score	0.8448
Accuracy	0.8453

Classification Report SVM

	Precission	Recall	F1-Score	Support
Negative	0.77	0.77	0.77	172
Positive	0.89	0.89	0.89	371
Accuracy			0.85	543
Macro avg	0.83	0.83	0.83	543
Weighted avg	0.85	0.85	0.85	543

F1-Score	0.8544
Accuracy	0.8545

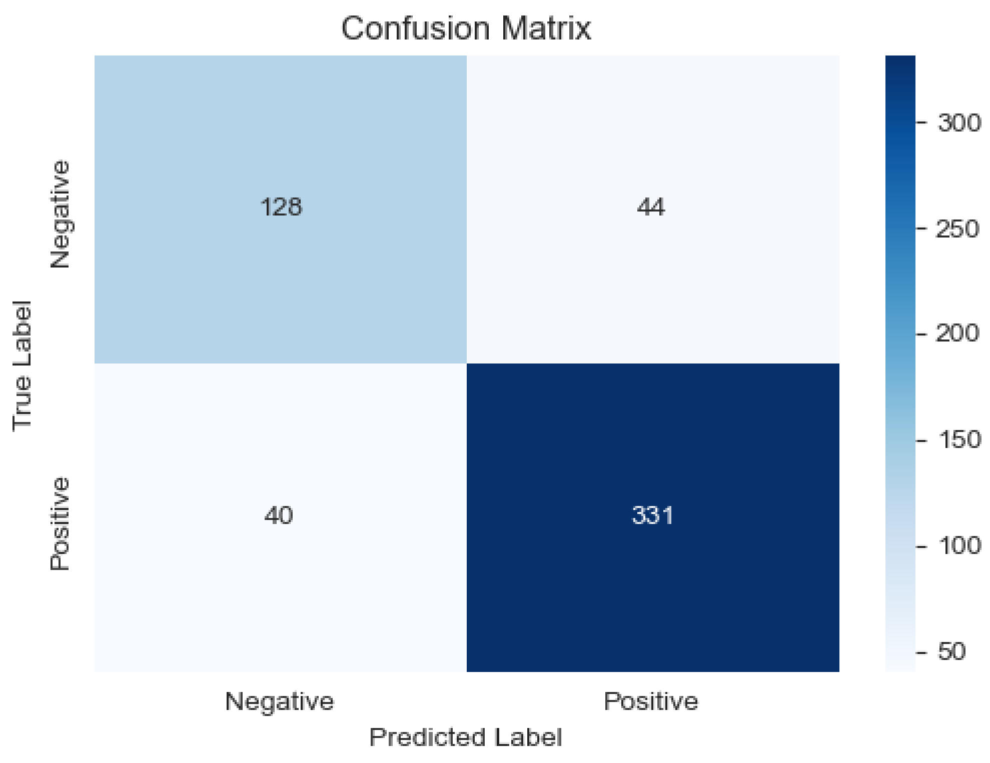

Image 3: Confusion Matrix Logistic Regression

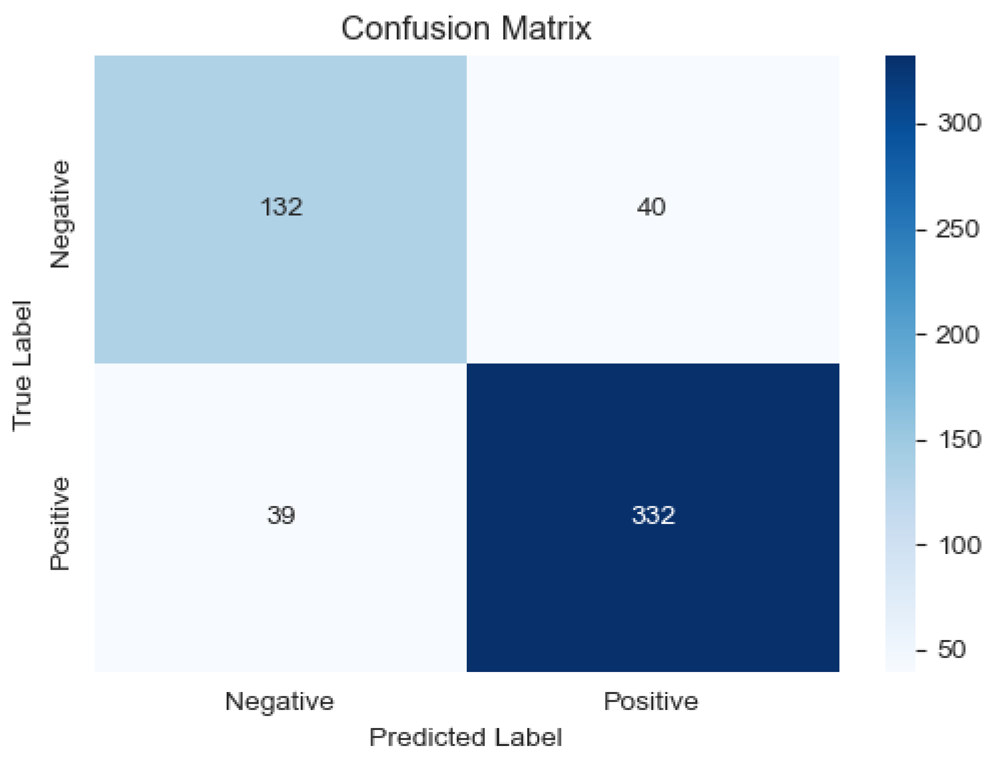

Image 4: Confusion Matrix SVM

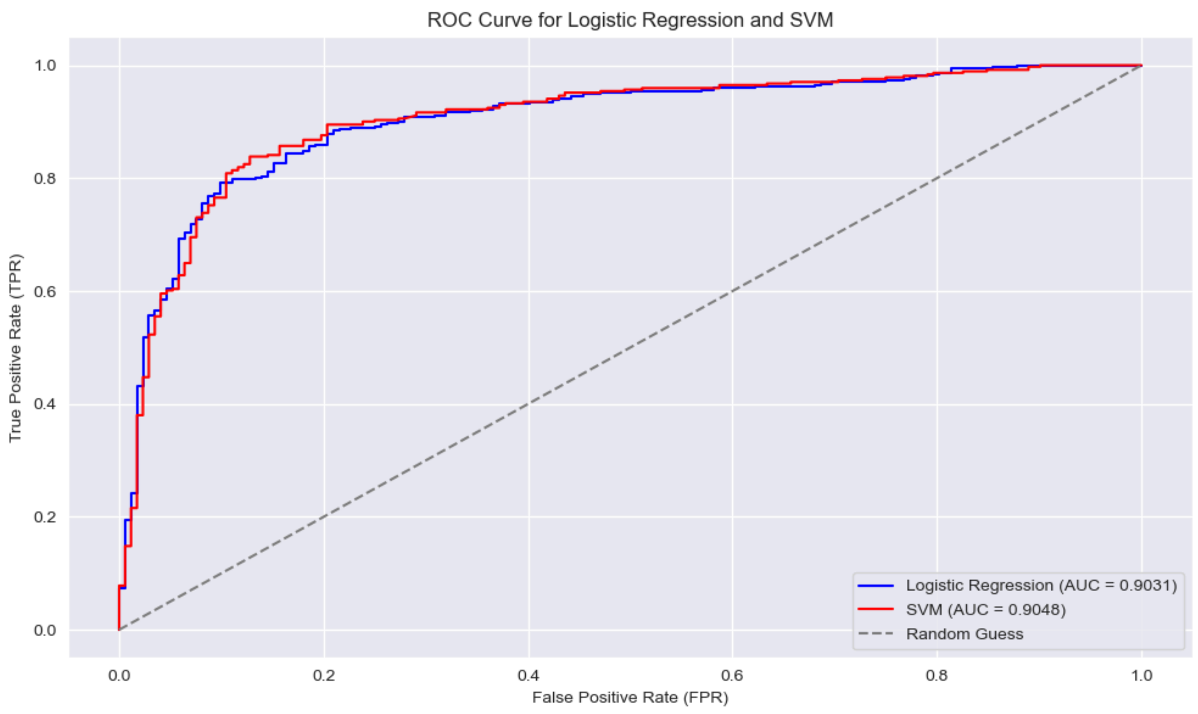

Image 5: ROC curves for baseline models

Looking at the results of the classification report for the logistic regression model, it is observed that of the 172 data that belonged to the negative class the model successfully identified 75% and of the 371 data that belonged to the positive class it successfully identified 89% of the data with an overall success in f1 score at 84,48% and in accuracy 84,53%.Also the AUC index has a success rate of 90,31% suggesting that the model is likely to generalize well to unseen data because it is robust across thresholds. On the other hand, the SVM had similar results to the logistic regression in the recognition of positive classes as it also recognized 89% but also performed much better in terms of recognition of negative classes as it recognized 77% of the 172 data correctly and also had equalized preccision and recall values. The AUC of the model is slightly higher than the logistic regression with a rate of 90.48% which indicates that this model is likely to generalize well to unseen data also. In terms of overall accuracy and f1 score the SVM has 85.45% and 85.44% respectively. As for the fact that both models have difficulty in identifying the negative class this may be due to the test data set as the data associated with the positive class are more and the data belonging to the negative class are less and it may be that the data that are negative are data that also have similar characteristics to the data associated with the positive class. One way to address this problem is to use crossvalidation and to train and test the models on different data sets. Lastly, it should be noted that the reason that both models have almost the same results is that they are both effective linear classifiers (the SVM depending on the kernel used can find non-linear classifiers). As in this particular implementation the svm is built to use the linear kernel and therefore the results are quite close.

Deep Learning Models

The models used are the LSTM and CNN models and the reason why they were used is that both models are ideal for tasks such as sentiment analysis and excel in this area. The Lstm in particular is ideal in capturing the context and dependencies between words in a sentence as it is designed to handle sequential data. CNN excel at identifying patterns such as phrases or key n-grams in text, using convolutional filters to extract features efficiently and are also quite effective in retaining the information extracted from the text.

Model Implementation

LSTM :

1. **Frameworks and Libraries**:
    a. Pytorch
    b. sklearn,seaborn
2. **Hardware:** Model trained on GPU
3. **Dataset Preparation :**
    a. Dataset class was created to handle TF-IDF features and corresponding sentiment labels.
    b. Dataloader for batching and shuffling the data
4. **Model Architecture** :
    a. **Input Size**: Equal to the number of TF-IDF features.
    b. **Hidden Size**: Set to 256 neurons for the LSTM layer.
    c. **Number of Layers**: Two stacked LSTM layers.
    d. **Dropout**: Applied 30% dropout for regularization to prevent overfitting.
    e. **Fully Connected Layer**: Maps the last hidden state to the output classes (positive/negative sentiment).
5. **Hyperparameters :**
    a. Optimizer: **AdamW** with a learning rate of 0.001 and weight decay of 10−410^{-4}10−4.
    b. Loss Function: **CrossEntropyLoss** for binary classification.
    c. Scheduler: **ReduceLROnPlateau** to reduce the learning rate when validation loss plateaued.

    d. Batch Size: 32.
    e. Epochs: 5 (with early stopping).
6. **Regularization Techniques**:
    a. Early stopping implemented stops after 3 epochs if validation loss stops improving
    b. Dropout layers between LSTM layers and fully connected layer
7. **Training Process**:
    a. Loss and accuracy monitoring through tqdm
    b. Forward pass a sequence dimension to input TF-IDF features for compatibility with the LSTM.
    c. Backward pass use of loss.backward() to calculate gradient and weights update via optimizer.step()
8. **Validation Process**:
    a. Model evaluation after each epoch with the validation set
    b. Adjusted learning rate based on validation loss.

CNN :

1. **Frameworks and Libraries**:
    a. Pytorch
    b. sklearn,seaborn
2. **Hardware:** Model trained on GPU
3. **Dataset Preparation :**
    a. Dataset class was created to handle TF-IDF features and corresponding sentiment labels.
    b. Dataloader for batching and shuffling the data
    c. Data was reshaped to include a channel dimension to be compatible with the CNN layers
4. **Model Architecture**:
    a. Convolutional Layers :
        i. Layer 1: Input channels = TF-IDF feature size, output channels = 256, kernel size = 3.
        ii. Layer 2: Input channels = 256, output channels = 128, kernel size = 3.
    b. ReLU activation applied to introduce non-linearity.
    c. Gobal max pooling for size reduction for the feature map size to a fixed vector
    d. Fully connected layers:
        i. Layer 1: 128 input neurons mapped to 64 neurons with ReLU activation.
        ii. Layer 2: 64 neurons mapped to the output classes (positive/negative).
    e. Dropout: 30% dropout added to prevent overfitting
5. **Hyperparameters**:

   a. Loss Function: **CrossEntropyLoss** for binary sentiment classification.
   b. Optimizer: **AdamW** with a learning rate of 0.001 and weight decay of 10−410^{-4}10−4.
   c. Scheduler: **ReduceLROnPlateau** to reduce learning rate on validation loss plateau.
   d. Batch Size: 32.
   e. Epochs: 5 (with early stopping).
6. **Training Process**:
   a. Forward pass: Input reshaped to include a channel dimension, followed by convolution and pooling operations.
   b. Backward pass:
      i. CrosssEntropyLoss
      ii. AdamW
   c. tqdm for monitoring
7. **Validation Process**:
   a. Model evaluation after each epoch with the validation set
   b. Adjusted learning rate based on validation loss.

Deep learning models evaluation

The models were trained and tested also with the previous preprocessing techniques as mentioned earlier in the baseline models. Also for the evaluation of the models the same metrics will be used.

Classification Report LSTM

	Precission	Recall	F1-Score	Support
Negative	0.75	0.75	0.75	172
Positive	0.88	0.89	0.89	371
Accuracy			0.84	543
Macro avg	0.82	0.82	0.82	543
Weighted avg	0.84	0.84	0.84	543

F1-Score	0.8855
Accuracy	0.8434

Classification Report CNN

	Precission	Recall	F1-Score	Support
Negative	0.68	0.81	0.74	172
Positive	0.90	0.82	0.86	371
Accuracy			0.82	543
Macro avg	0.79	0.82	0.80	543
Weighted avg	0.83	0.82	0.82	543

F1-Score	0.8619
Accuracy	0.8195

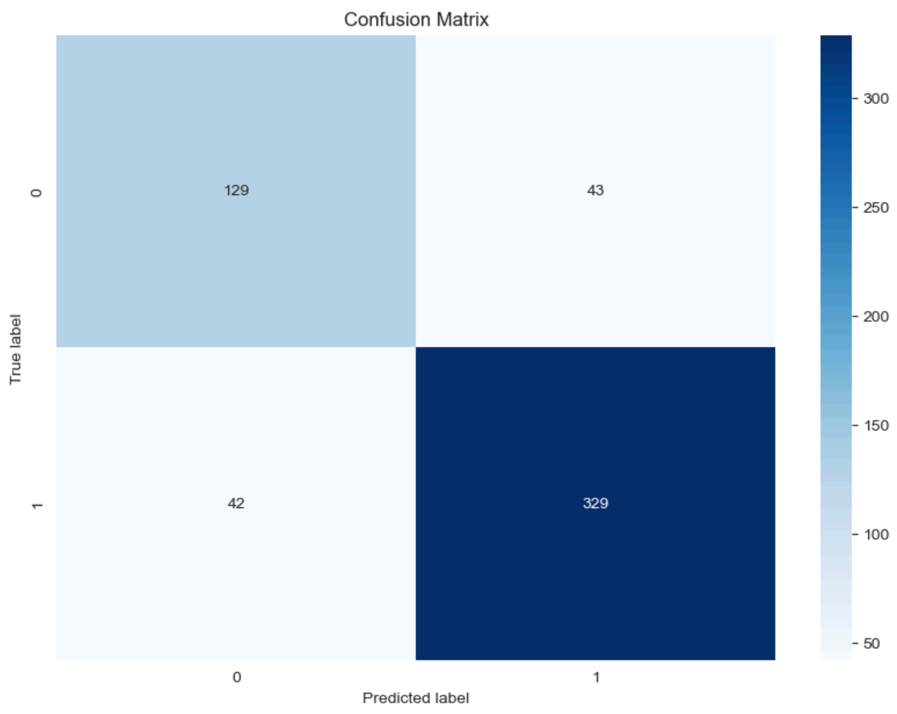

Image 5: Confusion Matrix LSTM

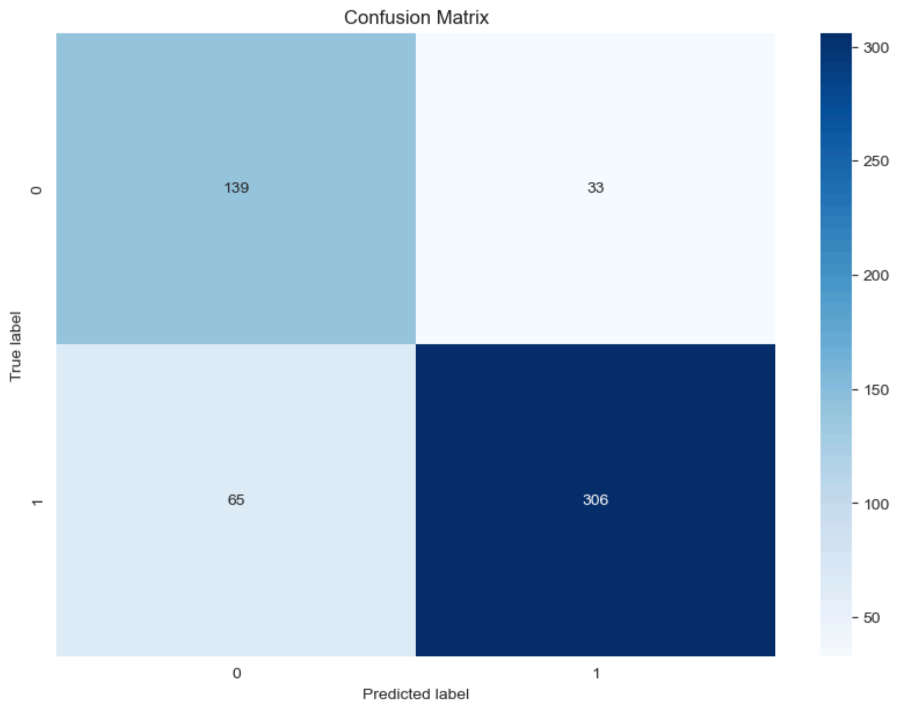

Image 6: Confusion Matrix CNN

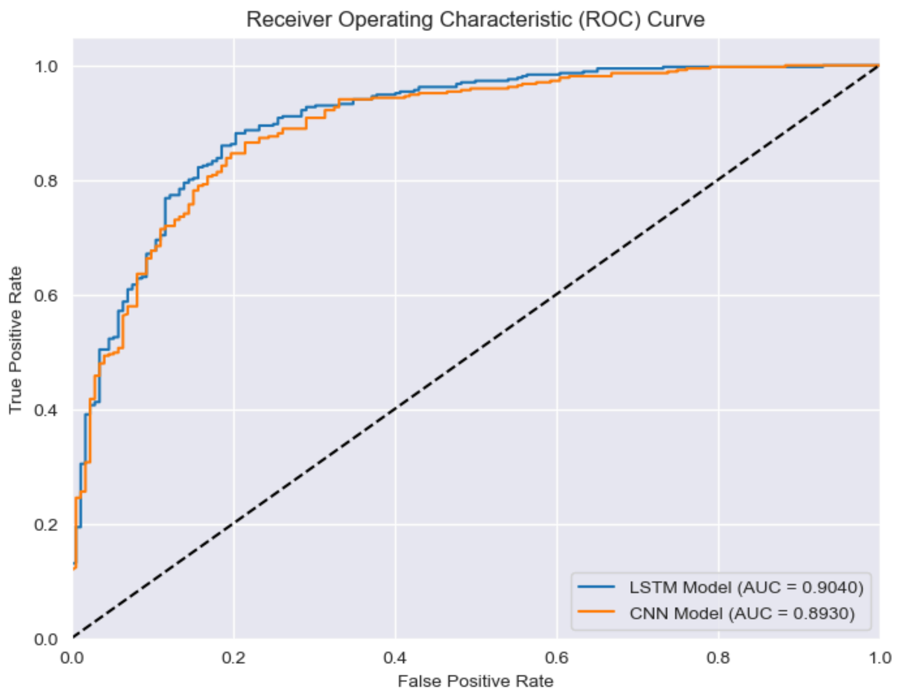

Image 7: ROC Curves

The LSTM model in terms of its overall performance gives us an accuracy of 84.34% and an F1 score of 88.55% and manages to successfully identify the data belonging to the positive class with an F1-Score of 89%. As for CNN its overall performance in accuracy and F1-Score is 81.95% and 86.19% respectively and in the positive class it has F1-Score at 86%. As for the negative class both models perform with 75% f1 score LSTM and CNN with 74% respectively . Looking at the ROC curves and AUC index LSTM has 90.40% and CNN 89.30% these results are encouraging as they can detect and distinguish the two classes quite well. The problem that arises here too is the difficulty in identifying the negative class which is also due in this case to the reasons previously mentioned in the evaluation of the baseline model. Overall both models performed well with this dataset.

5. Comparisons And Evaluation

The chapter compares the baseline model and the deep learning model on the strengths and limitations of the models based on their results on sentiment analysis in the previous chapter. Also in this chapter is presented the comparison between the different preprocessing techniques.

Model Comparison

The tables below contain all the results from the models that were trained and compare their performance overall(accuracy,f1-score ,AUC) and how effective they were in identifying classes.

Model Comparison using overall accuracy, F1-Score and AUC

Statistics	Logistic Regression	SVM	LSTM	CNN
F1-Score	84,48%	85,44%	88,55%	86,19%
Accuracy	84,53%	85,45%	84,34%	81,95%
AUC	90.31%	90.48%	90,40%	89,30%

Model Comparison for class recognition(f1-score)

Classes	Logistic Regression	SVM	LSTM	CNN
Negative	75%	77%	75%	74%
Possitive	89%	89%	89%	86%

Having taken into account the data from the two tables, the following conclusions can be drawn. In overall performance LSTM has the highest F1-Score indicating that it has better balance between precision and recall across classes. SVM has the highest accuracy and AUC making it highly competitive for this task. CNN has the lowest accuracy but maintains a high F1-Score. As for the class recognition, all models perform better in finding the positive class with 89% F1-Score in LSTM, SVM and Logistic Regression. As for the recognition of the negative class varies slightly, SVM is showing the best F1-Score for this class. Based on these conclusions LSTM is a quite efficient method as it has the best overall results in all metrics and quite efficient in terms of managing both classes. Also quite good in its performance is SVM as it excels in Accuracy and in AUC and unlike all other models it is the model with the most success in identifying negative classes . Also SVM is a strong alternative if computational simplicity is prioritized.

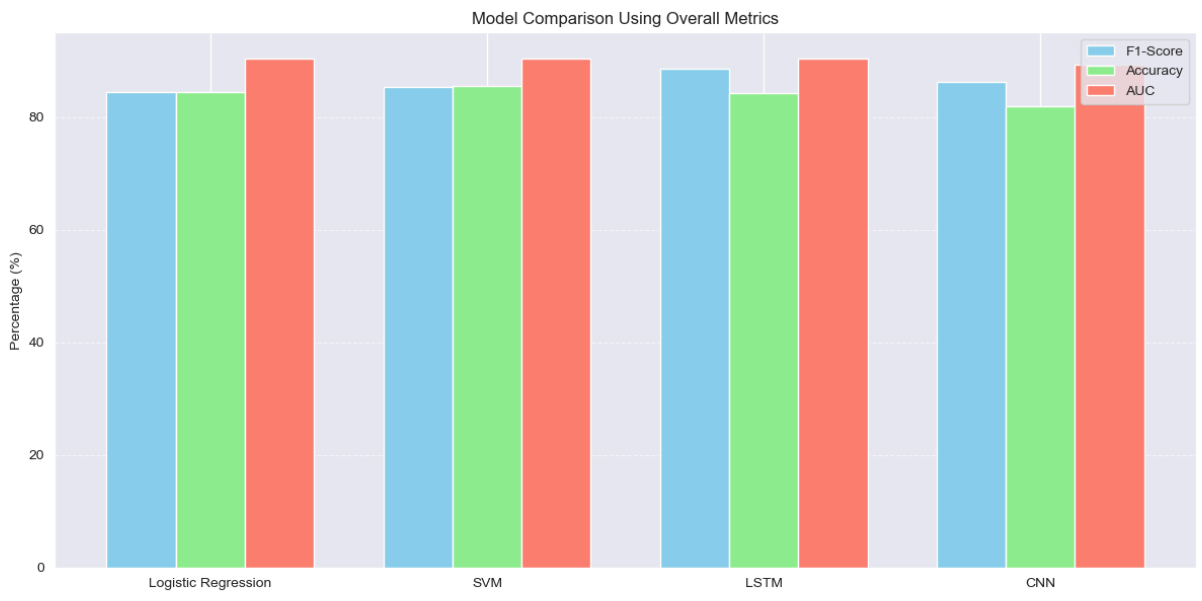

Image 8: Model comparison using overall metrics

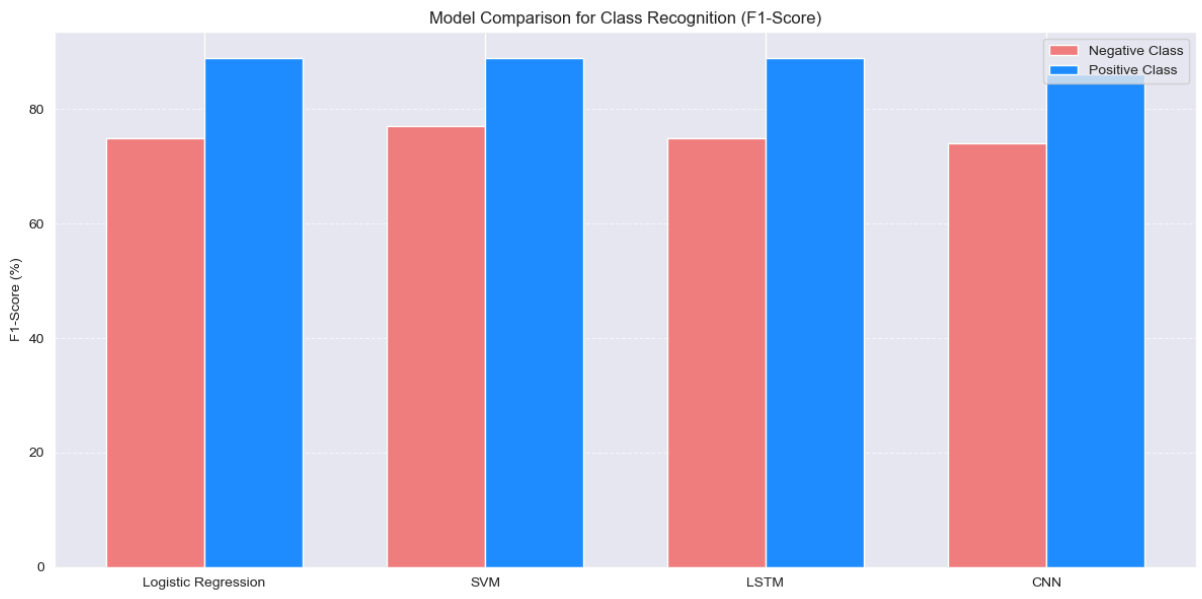

Image 9: Model Comparison for class recognition

Preprocessing Methods Comparison

This chapter will focus on how different preprocessing methods impact the results of the models that were shown previously. The methods that gonna be explore are :

- Stemming
- Stemming with stopwords
- Lemmatization
- Lemmatization with stopwords

Preprocessing Methods Comparison Table (f1-score)

Preprocessing Methods	Logistic Regression	SVM	LSTM	CNN
Stemming	84,48%	85,44%	88,55%	86,19%
Stemming with stopwords	81,55%	79,69%	84,71%	84,25%
Lemmatization	82,75%	83,75%	86.89%	86,24%
Lemmatization with stopwords	80.31%	80.31%	84,90%	84,45%

Observing the table with the results it is clear that removing stopwords and when stemming is used and when lemmatization is used the performance of all models drops. This is due to the fact that stopwords contain words that can be crucial for sentiment analysis, such as handling negations. As far as stemming is concerned, it is observed that it gives much better results than lemmatization as it is possible that removing the suffixes and converting the words to the root form significantly reduces the size of the vocabulary and leads to better generalization. It is notable that deep learning models in all cases outperform machine learning models. Also the combination of lemmatization and stopwords removal gives much better results when stopwords are removed when stemming is used. Also deep learning models show greater tolerance when stopwords are removed as opposed to machine learning models. In summary the usage of stemming in this particular dataset tends to give better performance in classification than the other preprocessing methods.

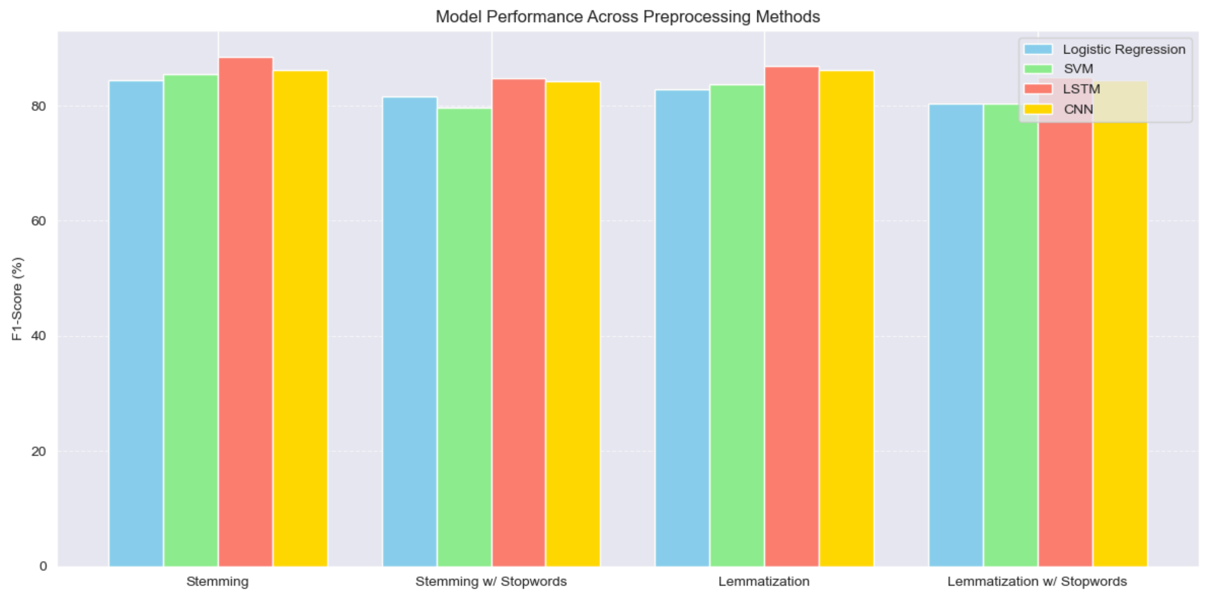

Image 10: Model Performance Across Preprocessing Methods

Diffrences Between Baseline and Deep Learning Models

Baseline models are easier to use and implement than the deep learning models. Baseline models are computational lightweight in comparison with the deep learning models and that makes them suitable for smaller datasets due to the less time they need to train. On the other hand deep learning models may need more computational power but their performance and the longer training time they need makes them better for larger datasets and for dealing with more complex sentiment analysis tasks offering better scalability and generalization. The deep learning models are also resilient in different preprocessing methods as was shown previously in comparison with the baseline models which can significantly impact their performance. In conclusion, baseline models are ideal for tasks that do not require a lot of computational resources and simpler tasks while deep learning models are ideally suited for detailed and nuanced sentiment analysis in larger, complex datasets.

6. Conclusions

This project explored the use of sentiment analysis on the financial domain using machine learning(Logistic Regression,SVM) and deep learning models(LSTM,CNN). The results of this research showed that the deep learning models and in particular the LSTM model performed the best with F1-Score (88.55%) and AUC (90.40%). As for the preprocessing part, it clearly affects the performance of the models as the use of stemming provided better results than lemmatization. Also the removal of stopwords reduced performance, suggesting the importance of contextual cues provided by stopwords. Overall, the findings of this work show the importance of using correct preprocessing methods and also the importance of selecting the appropriate model to be used for domain specific sentiment analysis.

7. Future Work

Future plans are to test models such as distilBert and Bert as well as to look for other preprocessing methods such as sentiment-aware embeddings or domain-specific lexicons. As well as the use of other feature extraction methods such as word embeddings and Incorporate more diverse financial datasets to improve model generalization and also including the neutral sentiment for exploring multiclass classification tasks.

Appendix

1. Tables and metrics

Preprocessing Methods Comparison Table

Preprocessing methods	Models	F1-Score	Accuracy	AUC
Stemming	Logistic Regresion	84,48%	84,53%	90,30%
	SVM	85,45%	85,45%	90,48%
	LSTM	88,55%	84,34%	90,40%
	CNN	86,19%	81,95%	89,30%
Stemming Stopwords	Logistic Regresion	81,55%	81,58%	87,44%
	SVM	79,69%	79,56%	86,60%
	LSTM	84,71%	79,00%	86,59%
	CNN	84,25%	78,45%	85,89%
Lemmatization	Logistic Regresion	82,75%	82,87%	88,88%
	SVM	83,77%	83,79%	88,71%
	LSTM	86,89%	81,95%	89,07%
	CNN	86,24%	80,84%	87,42%
Lemmatization with Stopwords	Logistic Regresion	80,31%	80,29%	85,90%
	SVM	80,31%	80,11%	84,83%
	LSTM	84.90%	79,37%	85,79%
	CNN	84,45%	78,63%	84,45%

Classification Report Logistic Regression (Stemming)

	Precission	Recall	F1-Score	Support
Negative	0.76	0.74	0.75	172
Positive	0.88	0.89	0.89	371
Accuracy			0.85	543
Macro avg	0.82	0.82	0.82	543
Weighted avg	0.84	0.85	0.85	543

Classification Report SVM (Stemming)

	Precission	Recall	F1-Score	Support
Negative	0.77	0.77	0.77	172
Positive	0.89	0.89	0.89	371
Accuracy			0.85	543
Macro avg	0.83	0.83	0.83	543
Weighted avg	0.85	0.85	0.85	543

Classification Report LSTM (Stemming)

	Precission	Recall	F1-Score	Support
Negative	0.75	0.75	0.75	172
Positive	0.88	0.89	0.89	371
Accuracy			0.84	543
Macro avg	0.82	0.82	0.82	543
Weighted avg	0.84	0.84	0.84	543

Classification Report CNN (Stemming)

	Precission	Recall	F1-Score	Support
Negative	0.68	0.81	0.74	172
Positive	0.90	0.82	0.86	371
Accuracy			0.82	543
Macro avg	0.79	0.82	0.80	543
Weighted avg	0.83	0.82	0.82	543

Classification Report Logistic Regression(Stemming Stopwords)

	Precission	Recall	F1-Score	Support
Negative	0.71	0.70	0.71	172
Positive	0.87	0.87	0.87	371
Accuracy			0.83	543
Macro avg	0.79	0.79	0.79	543
Weighted avg	0.82	0.82	0.82	543

Classification Report SVM (Stemming Stopwords)

	Precission	Recall	F1-Score	Support
Negative	0.67	0.70	0.69	172
Positive	0.86	0.84	0.85	371
Accuracy			0.80	543
Macro avg	0.76	0.77	0.77	543
Weighted avg	0.80	0.80	0.80	543

Classification Report LSTM (Stemming Stopwords)

	Precission	Recall	F1-Score	Support
Negative	0.67	0.66	0.75	172
Positive	0.84	0.85	0.85	371
Accuracy			0.79	543
Macro avg	0.76	0.75	0.76	543
Weighted avg	0.79	0.79	0.79	543

Classification Report CNN (Stemming Stopwords)

	Precission	Recall	F1-Score	Support
Negative	0.66	0.66	0.66	172
Positive	0.84	0.84	0.84	371
Accuracy			0.78	543
Macro avg	0.75	0.75	0.75	543
Weighted avg	0.78	0.78	0.78	543

Classification Report Logistic Regression (Lemmitazation)

	Precission	Recall	F1-Score	Support
Negative	0.74	0.70	0.72	172
Positive	0.87	0.89	0.88	371
Accuracy			0.83	543
Macro avg	0.80	0.80	0.80	543
Weighted avg	0.83	0.83	0.83	543

Classification Report SVM (Lemmatization)

	Precission	Recall	F1-Score	Support
Negative	0.75	0.74	0.74	172
Positive	0.88	0.88	0.88	371
Accuracy			0.84	543
Macro avg	0.81	0.81	0.81	543
Weighted avg	0.84	0.84	0.84	543

Classification Report LSTM (Lemmatiazation)

	Precission	Recall	F1-Score	Support
Negative	0.72	0.70	0.71	172
Positive	0.86	0.88	0.87	371
Accuracy			0.82	543
Macro avg	0.79	0.79	0.79	543
Weighted avg	0.82	0.82	0.82	543

Classification Report CNN (Lemmatization)

	Precission	Recall	F1-Score	Support
Negative	0.72	0.66	0.68	172
Positive	0.85	0.88	0.86	371
Accuracy			0.81	543
Macro avg	0.78	0.77	0.77	543
Weighted avg	0.81	0.81	0.81	543

Classification Report Logistic Regression (Lemmatization Stopwords)

	Precission	Recall	F1-Score	Support
Negative	0.69	0.69	0.69	172
Positive	0.86	0.85	0.86	371
Accuracy			0.80	543
Macro avg	0.77	0.77	0.77	543
Weighted avg	0.80	0.80	0.80	543

Classification Report SVM (Lemmatization Stopwords)

	Precission	Recall	F1-Score	Support
Negative	0.67	0.73	0.70	172
Positive	0.87	0.84	0.85	371
Accuracy			0.80	543
Macro avg	0.77	0.78	0.77	543
Weighted avg	0.81	0.80	0.80	543

Classification Report LSTM (Lemmatization Stopwords)

	Precission	Recall	F1-Score	Support
Negative	0.67	0.67	0.67	172
Positive	0.85	0.85	0.85	371
Accuracy			0.79	543
Macro avg	0.76	0.76	0.76	543
Weighted avg	0.79	0.79	0.79	543

Classification Report CNN (Lemmatization Stopwords)

	Precission	Recall	F1-Score	Support
Negative	0.67	0.65	0.66	172
Positive	0.84	0.85	0.84	371
Accuracy			0.79	543
Macro avg	0.75	0.75	0.75	543
Weighted avg	0.79	0.79	0.79	543

2. Libraries and Tools

- **Python 3.12.8** : Programming language used for all implementations.
- **Cuda 12.4** : Drivers for utilizing GPU
- **collections**: Used Counter for counting elements.
- **html**: No specific tools used.
- **imblearn**: Used SMOTE for handling class imbalance.
- **jupyter_core**: Used pattern for notebook utilities.
- **matplotlib**: Used plt for plotting graphs.
- **nltk**: Used WordNetLemmatizer, stopwords, and word_tokenize for text preprocessing.
- **numpy**: Used np for numerical operations.
- **os**: No specific tools used.
- **pandas**: Used pd for data manipulation.
- **re**: No specific tools used.
- **seaborn**: Used sns for data visualization.
- **sklearn**: Used various tools for machine learning, including LabelEncoder, TfidfVectorizer, and performance metrics.
- **string**: No specific tools used.
- **torch**: Used DataLoader, Dataset, F, ReduceLROnPlateau, and nn for deep learning tasks.
- **tqdm**: Used tqdm for progress bars.

3. Links

Dataset: Financial Sentiment Analysis
Code: Sentiment-Analysis