UNIVERSITY PARTNER

UNIVERSITY OF WOLVERHAMPTON

HERALD COLLEGE KATHMANDU

# Text Classification

<Twitter sentimental analysis>

| | |
|---|---|
| University Id: | 2227738 |
| Name: | Nishant Niraula |
| Group: | L6CG2 |
| Lecturer: | Siman Giri |
| Tutor: | Sunita Parajuli |

## Table of Contents

**Table of Figure**

## 1. Introduction

- Classification of text fall in the category of NLP (Natural Language Processing) where the main goal is to classify the text data into already defined classes. In the text classification there are different analyzation and textual data understanding. Text classification is used in various purposes such as spam detection, sentimental analysis, content filtering etc. The Text classification we group members have chosen in our project is Twitter sentimental analysis. In this classification we classify the user sentiment base on the text that they have posted in the twitter application. The classes are divided into 3 categories. The classes are: Positive, Negative and Neutral. The main aim or objective of this classification is the classify the classes based on texts written by the users. Either it can be positive, negative or neutral. Also it provides the insights from the dataset that can be used in analysis field. This can help firm improve their product or services according to the demand of the user. (Kumar, 2023)

- Firstly, we initialize the sequential model. After initializing the model, we add the model embedding layer to our textual words into vector forms. This layer takes two parameters. The first one is Input dimension and second one is Output Dimension the Input dimension takes the size of the vocabulary and Output Dimension takes the dimension of the vector that we transformed from the textual words. We then move forward on adding the LSTM (Long Short Term Memory) layer with 128 hidden layer units. In LSTM layer we use return sequence true as it displays our result in every time step rather than only the final output. This allows our coming LSTM model to learn more long-term dependencies. Again we add other LSTM layer to capture long-term dependencies. In the last layer we add dense layer with the activation function of softmax which helps in our classification of multi class classification text. Also we use sparse categorical crossentropy as a loss function as it is widely used in multi-class classification. Also we use accuracy metrics to measure the performance of the model.

## 2. Methodology

### Data cleaning

- There are several steps that are involved in data cleaning process. Fistly the testing and training data are converted into the code reading form through pandas from csv file and columns names are set. The unnecessary columns are then dropped which is tweet ID and Entity. Then null of missing values are removed. Then duplicated rows are removed. Implementing these things results in cleaned dataset stored in the twitter_sentiments variable. Now the length of each tweet is calculated. Then mapping of the sentiment label of lower case value is done with uppercase value is done. (MarkovML, 2023)

### Model Architecture

- At first the model consists of embedding layer which is followed by LSTM layer which has 128 hidden layer unit and return sequence is set true. Followed by other LSTM layer with 128 hidden layer units. And lastly the dense layer which is also our output layer which is set to softmax activation function. Number of unique words is set to the input dimension of the embedding layer. Lastly the output dimension is set to 100 in the embedding layer.

## Hyper parameters

- There are different hyper parameters used while training the model. The hyper parameter includes:

Validation-split: 0.2

Epochs:15

LSTM layer units:128

Embedding the output dimensions: 100

The window size is set to 5 and minimum cost count is 1 and rate of the sample is 1e-3 while training from the Word2Vec NLTK library.

## 2.1 Data preprocessing

- There are several functions to preprocess the data. The first one is clean_text. In this function several cleaning process is done which includes lowercasing, removing unwanted tags, mentions etc.  And other functions such as remove_urls, remove_html_tags are used to perform cleaning that are used in clean_text function. The other function tokenize_and_lemmatize perform tokenization action and lemmatize every token into WordNetLemmitizer from the NLTK library. I have also used wordcloud to display the visual representation of the frequency of the words. (Erb, 2023) The word cloud that has been displayed in the training data is given below:

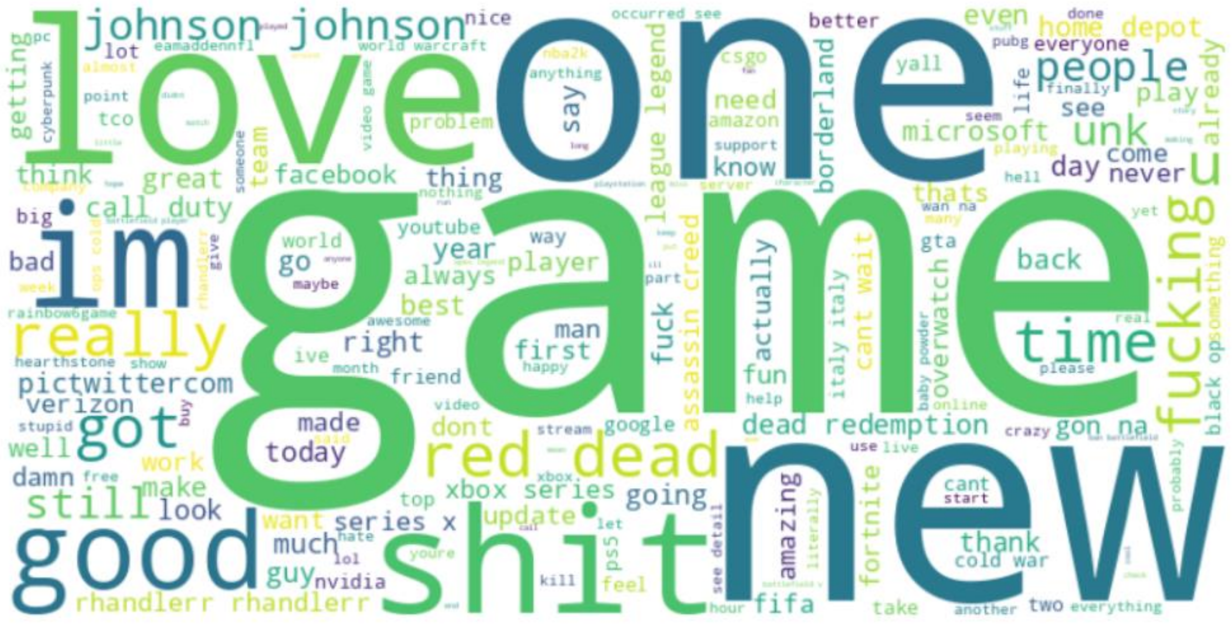


*Figure 1: Word Cloud*

## 2.2 Model Building

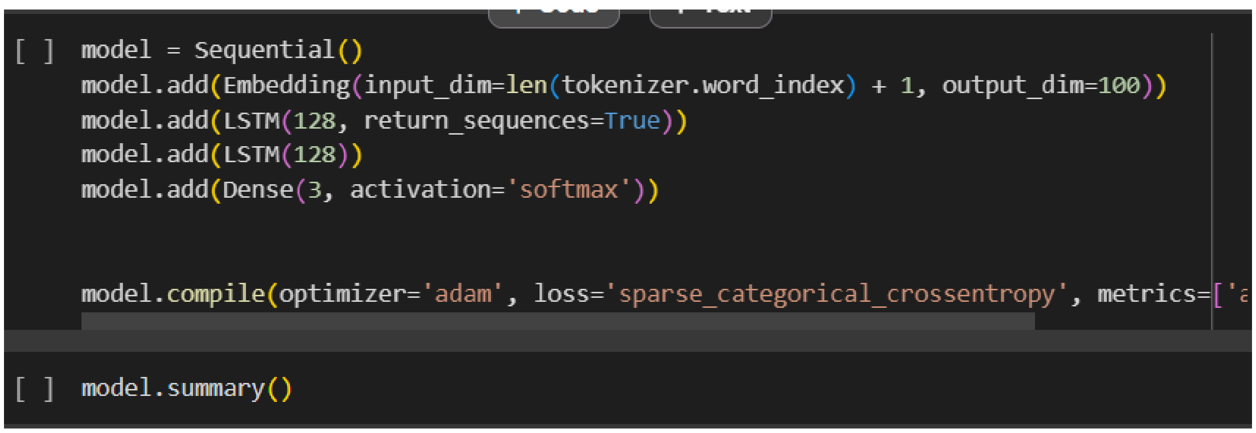
### Model Summary

- There consist of 3 layers while building this model: an embedding layer, LSTM layer, dense layer. Firstly, we initialize the sequential model. After initializing the model, we add the model embedding layer to our textual words into vector forms. This layer takes two parameters. The first one is Input dimension and second one is Output Dimension the Input dimension takes the size of the vocabulary and

Output Dimension takes the dimension of the vector that we transformed from the textual words. We then move forward on adding the LSTM (Long Short Term Memory) layer with 128 hidden layer units. In LSTM layer we use return sequence true as it displays our result in every time step rather than only the final output. This allows our coming LSTM model to learn more long-term dependencies. Again we add other LSTM layer to capture long-term dependencies. In the last layer we add dense layer with the activation function of softmax which helps in our classification of multi class classification text.

## Training of the model

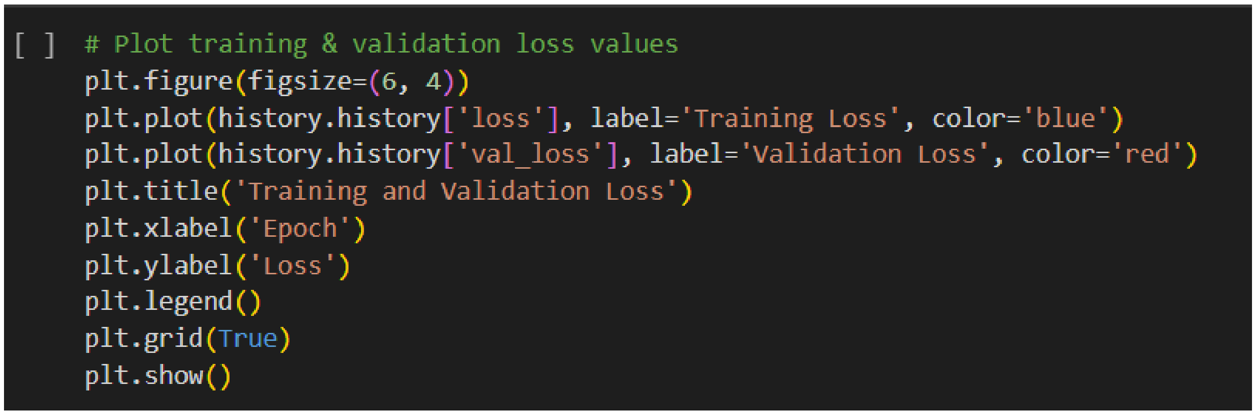
- For the training of the model the loss function that was used was ‘Sparse categorical cross-entropy’ and the optimizer that was used for optimizing the model was ‘Adam Optimizer’ and lastly 15 epochs was used for training the model. We have not increased the epochs as 15 epochs showed the desired result.



```
[ ]  model = Sequential()
     model.add(Embedding(input_dim=len(tokenizer.word_index) + 1, output_dim=100))
     model.add(LSTM(128, return_sequences=True))
     model.add(LSTM(128))
     model.add(Dense(3, activation='softmax'))


     model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['a

[ ]  model.summary()
```

*Figure 2:Building the model*

To specify the optimizer and loss function and metrics compile method is called. The fit methods train the training data. The validation split is of 0.2 which splits the 20% of our training data for validation.



```
[ ]  # Plot training & validation loss values
     plt.figure(figsize=(6, 4))
     plt.plot(history.history['loss'], label='Training Loss', color='blue')
     plt.plot(history.history['val_loss'], label='Validation Loss', color='red')
     plt.title('Training and Validation Loss')
     plt.xlabel('Epoch')
     plt.ylabel('Loss')
     plt.legend()
     plt.grid(True)
     plt.show()
```

*Figure 3:Plotting the result*

To extract the necessary curves like training and validation curve we are using history object that is coming from the fit method. And using matplotlib library to plot the desired output below.

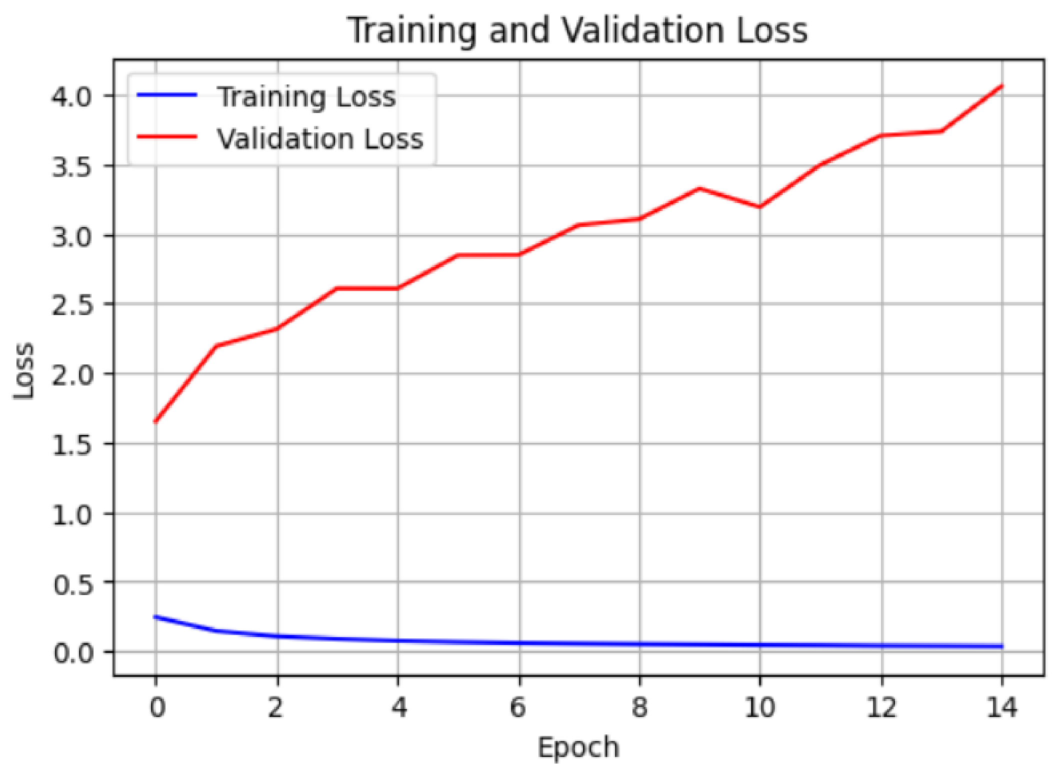
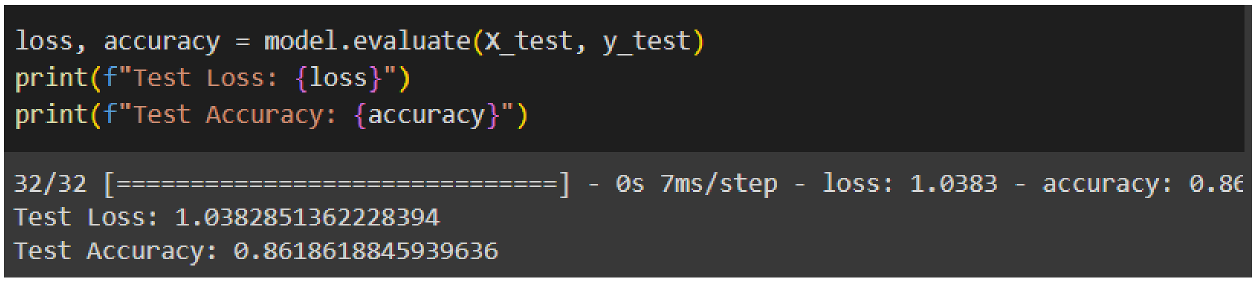


*Figure 4:Result*

From the above figure we can say that the training loss is decreasing over the increase of the epochs. While validation loss tends to increase over the increase of the epochs. Though these behaviors we can say that our model is over fitted. In summary, we can say that there is overfitting in the model and validation data tends to increase after 10th epochs.

**Evaluation of the model**

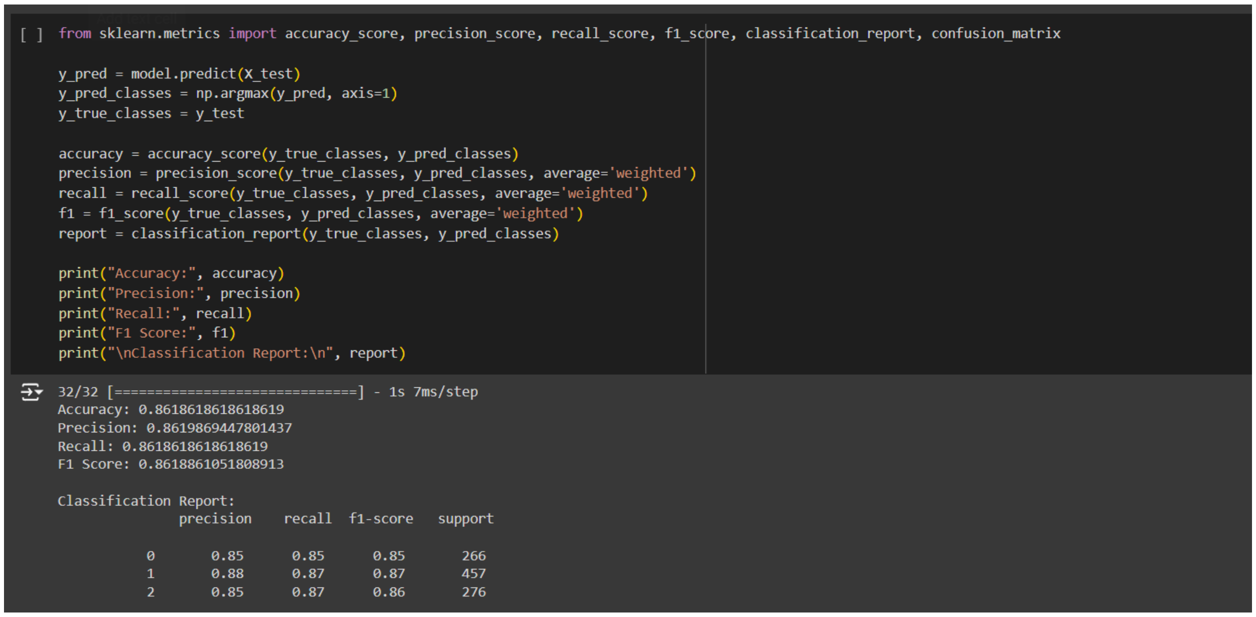


```
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Loss: {loss}")
print(f"Test Accuracy: {accuracy}")
```

```
32/32 [==============================] - 0s 7ms/step - loss: 1.0383 - accuracy: 0.86
Test Loss: 1.0382851362228394
Test Accuracy: 0.8618618845939636
```

*Figure 5: Test accuracy and Test loss*

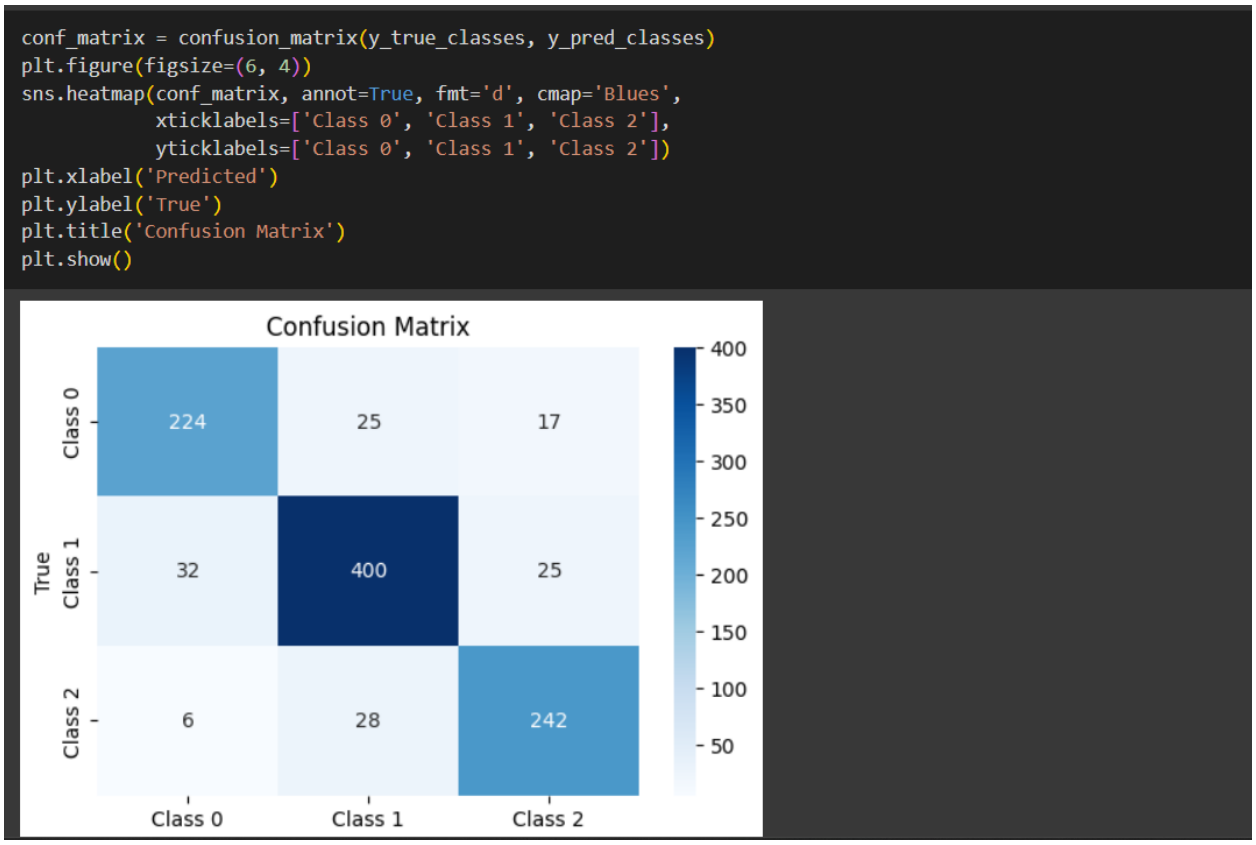
- The model has the training loss of 0.3212 and accuracy of training data of 0.8813 which indicates that our model has performed well in our prediction. The test loss is 1.038 and the accuracy of the test is 0.8619. These things indicates that model work best on training data rather than testing data. we can say that model may be overfitting in the training data.



```
[ ] from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report, confusion_matrix

    y_pred = model.predict(X_test)
    y_pred_classes = np.argmax(y_pred, axis=1)
    y_true_classes = y_test

    accuracy = accuracy_score(y_true_classes, y_pred_classes)
    precision = precision_score(y_true_classes, y_pred_classes, average='weighted')
    recall = recall_score(y_true_classes, y_pred_classes, average='weighted')
    f1 = f1_score(y_true_classes, y_pred_classes, average='weighted')
    report = classification_report(y_true_classes, y_pred_classes)

    print("Accuracy:", accuracy)
    print("Precision:", precision)
    print("Recall:", recall)
    print("F1 Score:", f1)
    print("\nClassification Report:\n", report)
```

```
32/32 [==============================] - 1s 7ms/step
Accuracy: 0.8618618618618619
Precision: 0.8619869447801437
Recall: 0.8618618618618619
F1 Score: 0.8618861051808913

Classification Report:
               precision    recall  f1-score   support

           0       0.85      0.85      0.85       266
           1       0.88      0.87      0.87       457
           2       0.85      0.87      0.86       276
```

*Figure 6:Evaluation metrics*

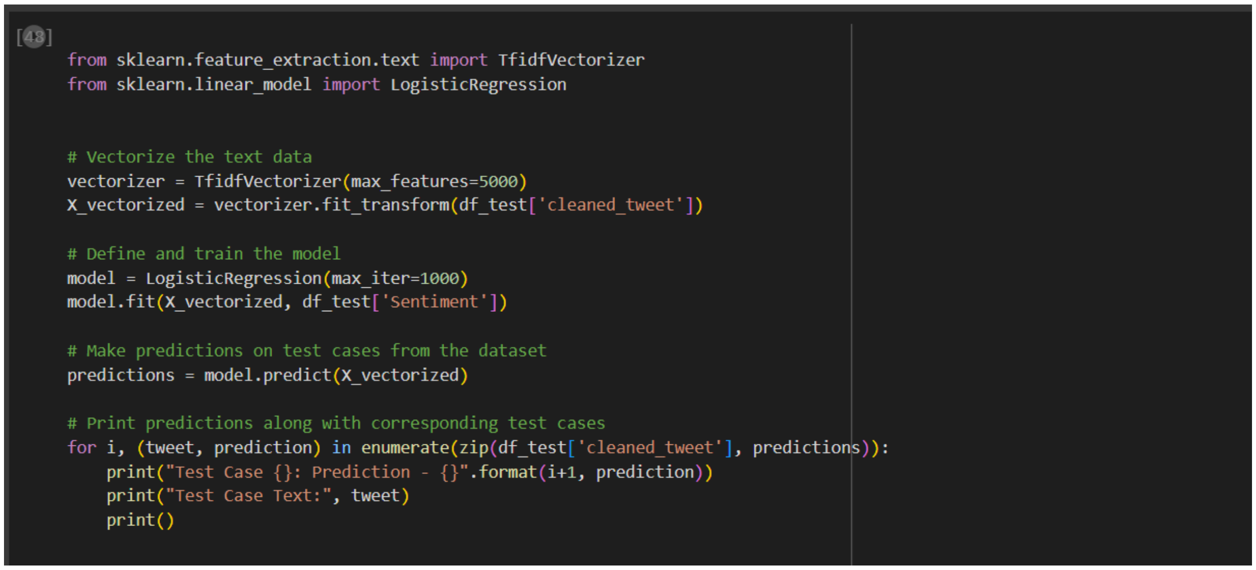


```
conf_matrix = confusion_matrix(y_true_classes, y_pred_classes)
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Class 0', 'Class 1', 'Class 2'],
            yticklabels=['Class 0', 'Class 1', 'Class 2'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

*Figure 7:Confusion Matrix*

Confusion matrix is analysis is done in this figure where there is True positive of 224 text of class 0. In class 1 there are 400 text truly predicted. And lastly in class 2 there are 242 texts truly predicted. So from here we can say that most of the text are truly predicted. So out accuracy is may be high.

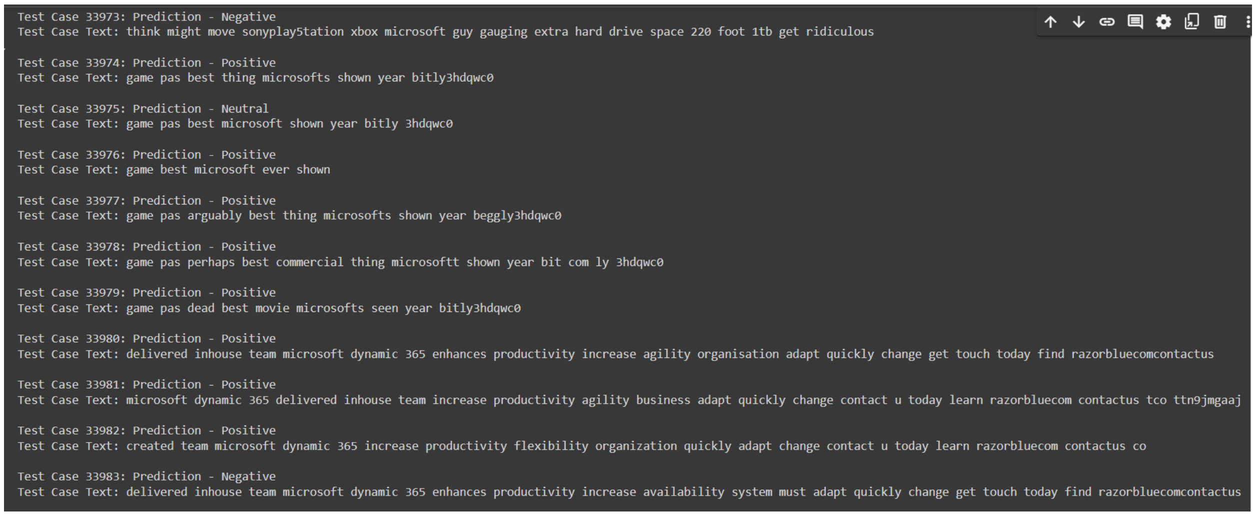
## Prediction in unseen data



```
[43]
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression


# Vectorize the text data
vectorizer = TfidfVectorizer(max_features=5000)
X_vectorized = vectorizer.fit_transform(df_test['cleaned_tweet'])

# Define and train the model
model = LogisticRegression(max_iter=1000)
model.fit(X_vectorized, df_test['Sentiment'])

# Make predictions on test cases from the dataset
predictions = model.predict(X_vectorized)

# Print predictions along with corresponding test cases
for i, (tweet, prediction) in enumerate(zip(df_test['cleaned_tweet'], predictions)):
    print("Test Case {}: Prediction - {}".format(i+1, prediction))
    print("Test Case Text:", tweet)
    print()
```

*Figure 8:Prediction Code sentiment analysis*

As we can see that we had tested out data in the test data. for this we have used logistic regression model for analysis of sentiments of the tweets. Firstly, text data is vectorized and it is vectorized using TF-IDF and it trains the model on the on the test data and then lastly the prediction is made on the test data. The prediction of each sentiment are shown in the figure below. From the figure below we can say that we have classified the text data into the sentiments and the predictions are shown with the tweets of the firms/entit.



```
Test Case 33973: Prediction - Negative
Test Case Text: think might move sonyplay5tation xbox microsoft guy gauging extra hard drive space 220 foot 1tb get ridiculous

Test Case 33974: Prediction - Positive
Test Case Text: game pas best thing microsofts shown year bitly3hdqwc0

Test Case 33975: Prediction - Neutral
Test Case Text: game pas best microsoft shown year bitly 3hdqwc0

Test Case 33976: Prediction - Positive
Test Case Text: game best microsoft ever shown

Test Case 33977: Prediction - Positive
Test Case Text: game pas arguably best thing microsofts shown year beggly3hdqwc0

Test Case 33978: Prediction - Positive
Test Case Text: game pas perhaps best commercial thing microsoftt shown year bit com ly 3hdqwc0

Test Case 33979: Prediction - Positive
Test Case Text: game pas dead best movie microsofts seen year bitly3hdqwc0

Test Case 33980: Prediction - Positive
Test Case Text: delivered inhouse team microsoft dynamic 365 enhances productivity increase agility organisation adapt quickly change get touch today find razorbluecomcontactus

Test Case 33981: Prediction - Positive
Test Case Text: microsoft dynamic 365 delivered inhouse team increase productivity agility business adapt quickly change contact u today learn razorbluecom contactus tco ttn9jmgaaj

Test Case 33982: Prediction - Positive
Test Case Text: created team microsoft dynamic 365 increase productivity flexibility organization quickly adapt change contact u today learn razorbluecom contactus co

Test Case 33983: Prediction - Negative
Test Case Text: delivered inhouse team microsoft dynamic 365 enhances productivity increase availability system must adapt quickly change get touch today find razorbluecomcontactus
```

*Figure 9:Predicted result*

## 3. Final Discussion

- So there are total 31,962 tweets in our dataset where there is the training split of 25,348 tweets and 6,614 in the testing split. The most of the sentiment labels are either positive or negative and very small number of sentiment labels are neutral or irrelevant.. And there is the accuracy of 85.5% in the testing data which shows the model has learnt well in classifying the sentiment of the tweets.

## 4. References

Erb, M., 2023. *Pre-Processing Tweets for Sentiment Analysis* | *by Mike Erb* | *Analytics Vidhya* | *Medium.* [Online]
Available at: https://medium.com/analytics-vidhya/pre-processing-tweets-for-sentiment-analysis-a74deda9993e
[Accessed 10 5 2024].

Kumar, K. K., 2023. *Text Classification on Twitter Data Using Machine Learning Algorithm.* [Online]
Available at: https://www.ijraset.com/research-paper/text-classification-on-twitter-data-using-machine-learning-algorithm#:~:text=Machine%20learning%20techniques%20play%20a,classify%20tweets%20into%20different%20categories.
[Accessed 10 5 2024].

MarkovML, 2023. *MarkovML Solutions.* [Online]
Available at: https://www.markovml.com/blog/data-cleansing
[Accessed 10 5 2024].