The model chosen for analysing Twitter's dataset is a hybrid model which is built with CNN-LSTM. This hybrid model combines both Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) to improve its performance in analysing sentimental text data by understanding the sentiments found in the tweets.

CNNs are capable of distinguishing patterns which is suitable to identify the patterns which are found in the text data. It is done through the convolutional layers that analyse the text data by filtering the patterns. The CNN component is suitable for processing big amounts of data as it is computationally resource-friendly, therefore it is believed to be optimal for running real-time sentiment analysis for Twitter (Kamyab et al., 2021).

As for LSTM, it is used to handle sequential data by keeping a memory of the previous data. This enables the model to have long-term memory in the text which can be used to understand and interpret the contents of the sentiment analysis. Due to LSTM's sequential data, the components are able to analyse the sequence of words in the tweet which helps in improving the accuracy in the sentiment predictions (Behera et al., 2021).

Therefore, by combining CNN and LSTM together, the model is believed to perform better compared to using the models individually as this model can perform sequential understanding and feature extraction through understanding the patterns in the text. Based on research, this model has been applied in real-world scenario to analyse the sentiment for an airline in Twitter and has achieved an accuracy of 91.3% making it a good choice to propose for this study (Jain et al., 2021).

By comparing the difference of Logistic Regression, SVM, and hybrid CNN-LSTM, it is observed that Logistic Regression is a linear model which predicts the binary outcome based on the probability of the input features. In this study, TF-IDF vectorization is used as the feature of the model. Therefore, there might be a challenge to understand the full language context. Support Vector Machine (SVM) in this study is a linear kernel model. It analyses and obtains the best line to separate the classes for classification, however it might not perform well with complicated and non-linear data. Furthermore, for the hybrid model which is CNN-LSTM, the model uses optimizers to adjust the weights, this will reduce the losses. Since the model is a hybrid, it consists of complex and non-convex optimisation landscape with regularization from the LSTM part of the model to prevent overfitting.

This section talks about Question 2 which is to compare and contrast the hybrid model, Logistic Regression and SVM.

(a) Architecture Comparison

The logistic regression model is made of a linear and simple architecture which is made of a single layer with features combined in that one layer. Hence, it struggles to handle complicated features. This is due to the model relies on pre-defined features which might not capture natural language as resourcefully as deep learning models (Zhang & Song, 2022).

The SVM model on the other hand, the architecture is linear kernel. It ultilises the margin between the classes for better classification. The linear kernel is effective in handling high volume data and it is computationally efficient (Ronran et al., 2020). However, it struggles in non-linear data.

Whereas for the hybrid model, the CNN-LSTM is made of many layers in the architecture. The CNN layer is in charge of extracting the features from the patterns and the LSTM layer is responsible for modelling the sequence in the patterns. This allows the model to obtain the local and global needs from the data (Al-Abyadh et al., 2022). This is due to the model can learn the features during training, while Logistic Regression and SVM only function on linear and pre-defined features.

(b) Performance Analysis

Based on performance, the logistic regression functions well in linear data, but its text data is often unstructured and contains non-linear patterns which can be challenging for logistic regression. However, in the aspect of efficiency, it is efficient in using computational resources, making it suitable to run real-time analysis with datasets at ease (Zhang & Song, 2022).

As for SVM, it is built similarly to logistic regression as it is the linear kernel. Besides that, it has the same struggle as logistic regression due to the text data pattern being non-linear. However, SVM has an advantage in generalisation as it maximizes the margins. SVM can efficiently process large datasets due to the high-dimensional spaces however the linear architecture causes it to have a lack of capability in capturing complex patterns (Ronran et al., 2020).

Besides that, the hybrid model (CNN+LSTM) performs well in collecting the local and sequential text from the data, this gives it an advantage for analysing sentiments. The model can automatically learn features based on the text data and this improves the effectiveness of analysing the language and complex lingo that are used in text of the tweets. Due to the nature of having deep learning structure, the model can be high in computational resources compared to SVM and logistic regression. However, this can be overcome using GPUs so that it can run large-size data efficiently.

(c) Suitability Application

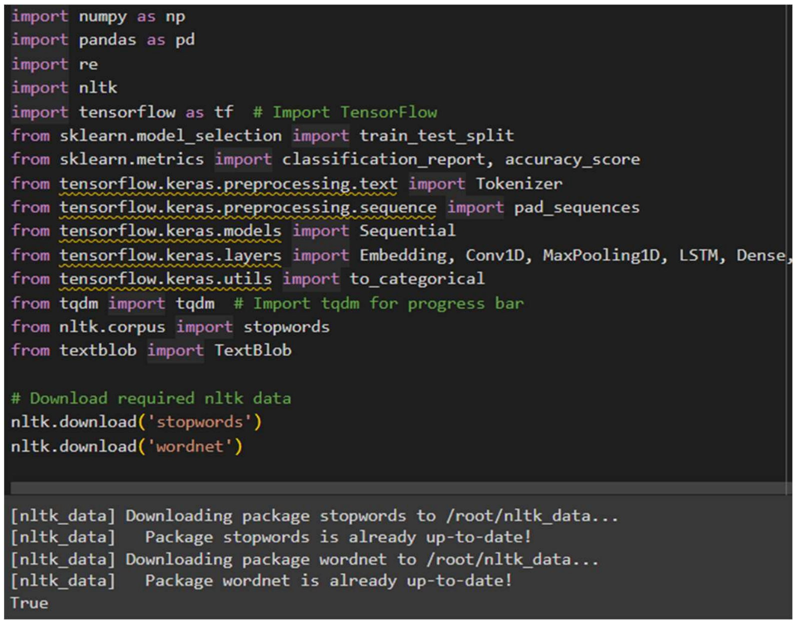
In the perspective of suitability in real-time sentiment analysis for large-scale Twitter data, logistic regression is easy to apply making it efficient to use for real-time analysis. However, if the data is large-scale like Twitter's data, it does not perform well due to the linear nature of the architecture and it relies on pre-defined engineering.

The SVM is suitable for funning large datasets for real-time processing as it can create linearly separable classes as is efficient in handling high dimensional data. However, the non-linear data such as text data will not be that efficient as the model is limited by it's linear built.

As for the hybrid model, it can be said that, it is suitable for large-scale data from Twitter as it can handle the sequential data and learn the complex features efficiently, with the combination of GPU, the optimization can be improve and therefore real-time analysis can be even more better as well. The disadvantage of the model is due to the need of high computational resources which can be energy and time consuming. It takes longer time to process which makes it a bit less efficient for running real-time applications.

Aspect	**Logistic Regression**	**SVM with Linear Kernel**	**Hybrid CNN-LSTM**
Architecture	- Linear model	- Linear model	- Multiple layers Model
	- Uses a sigmoid for mapping	- Uses linear for features	- CNN layers for extracting local patterns - LSTM layers for long-term dependencies
Feature Engineering	- Requires manual feature adjustment	- Requires manual feature adjustment	- Automatically learns features during training
Computational Efficiency	- Efficient for small to medium data	- Efficient unless big volume dataset	- Intensive due to deep structure
Performance on Complex Data	- Struggles with non-linear data	- Performs well for linearly separable data - Struggles with complicated patterns.	- Performs well in capturing non-linear relationships in data.
Suitability for Real-Time Processing	- Suitable due to the model is simple	- Suitable but slower than Logistic Regression when dealing large datasets.	- Suitable for real-time processing - Needs GPUs for large-scale data.
Scalability	- Good for small datasets - Might struggle with large datasets.	- Scalable for large datasets but high computation resources.	- Scales well with large datasets with use of GPU.

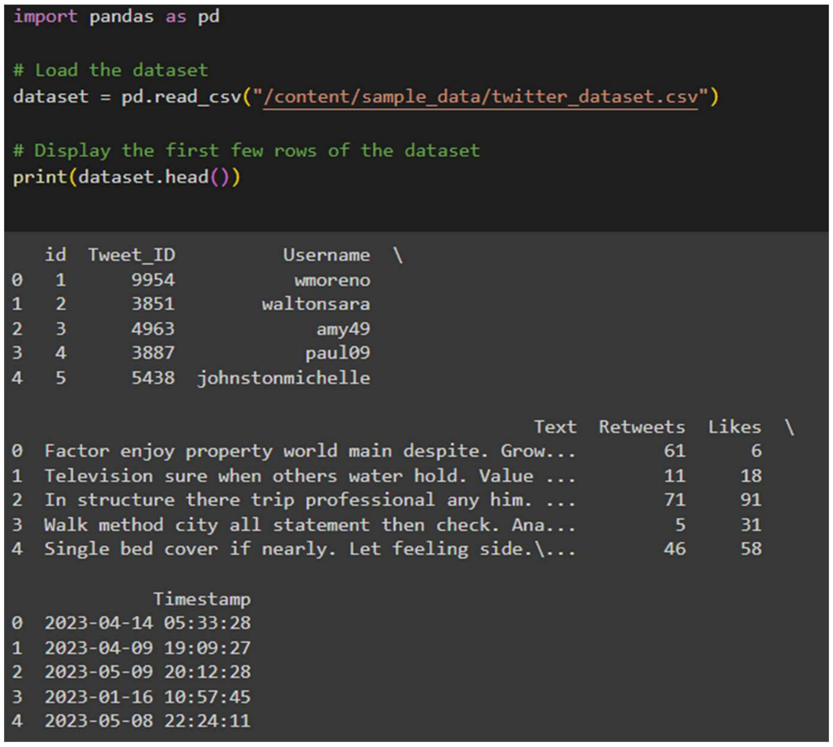
Before heading into Question 3, this section will be talking about how the dataset from Twitter is handled, the evaluation will be kept to Question 3.



```
import numpy as np
import pandas as pd
import re
import nltk
import tensorflow as tf  # Import TensorFlow
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Conv1D, MaxPooling1D, LSTM, Dense,
from tensorflow.keras.utils import to_categorical
from tqdm import tqdm  # Import tqdm for progress bar
from nltk.corpus import stopwords
from textblob import TextBlob

# Download required nltk data
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
True
```

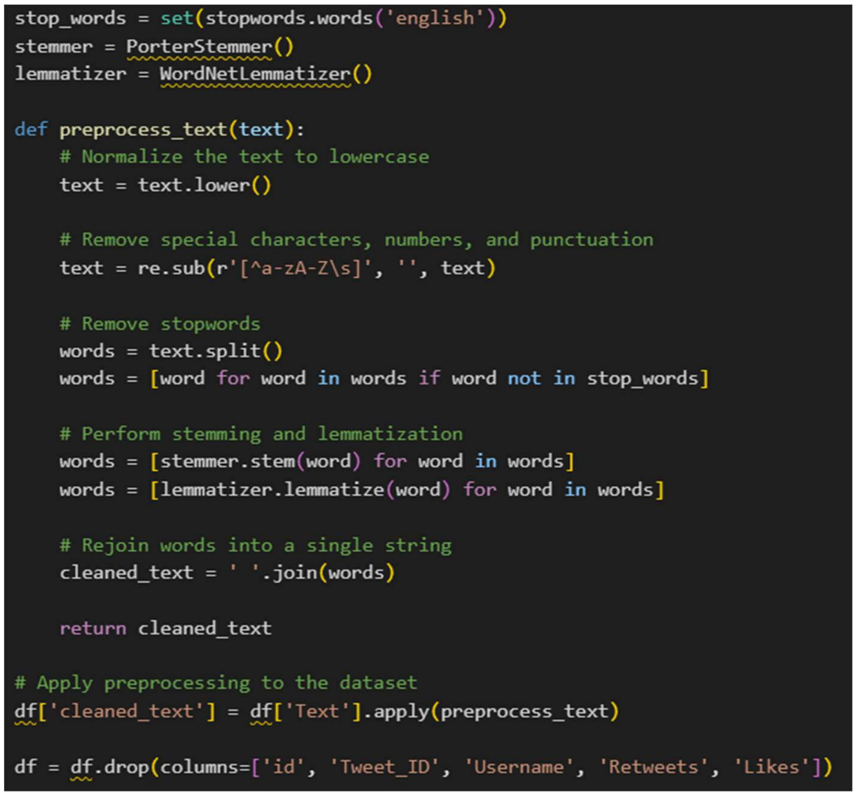
The libraries are imported into the python environment in Google Colab for further usage.



```
import pandas as pd

# Load the dataset
dataset = pd.read_csv("/content/sample_data/twitter_dataset.csv")

# Display the first few rows of the dataset
print(dataset.head())
```

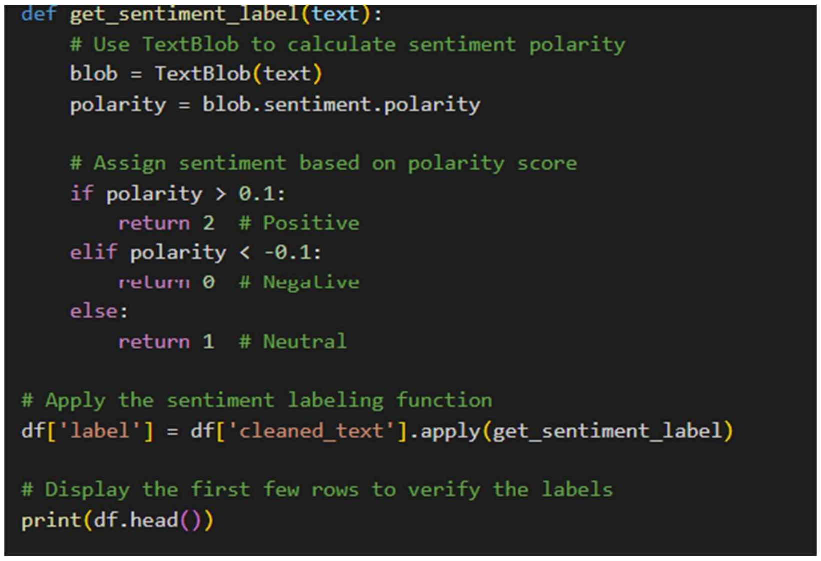
```
   id  Tweet_ID          Username  \
0   1      9954           wmoreno
1   2      3851        waltonsara
2   3      4963             amy49
3   4      3887            paul09
4   5      5438  johnstonmichelle

                                              Text  Retweets  Likes  \
0  Factor enjoy property world main despite. Grow...        61      6
1  Television sure when others water hold. Value ...        11     18
2  In structure there trip professional any him. ...        71     91
3  Walk method city all statement then check. Ana...         5     31
4  Single bed cover if nearly. Let feeling side.\...        46     58

             Timestamp
0  2023-04-14 05:33:28
1  2023-04-09 19:09:27
2  2023-05-09 20:12:28
3  2023-01-16 10:57:45
4  2023-05-08 22:24:11
```

The dataset is then imported and pandas is used here to read the csv file for Tweeter dataset, the first 5 rows of the columns are displayed to understand the structure of the dataset.



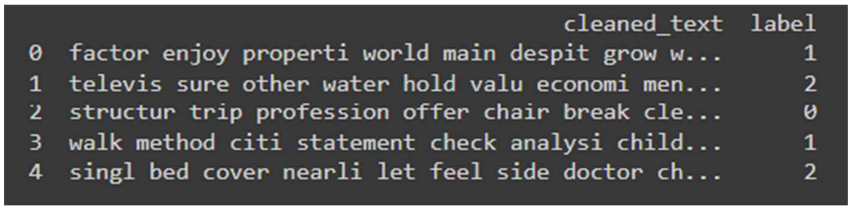
```
stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()

def preprocess_text(text):
    # Normalize the text to lowercase
    text = text.lower()

    # Remove special characters, numbers, and punctuation
    text = re.sub(r'[^a-zA-Z\s]', '', text)

    # Remove stopwords
    words = text.split()
    words = [word for word in words if word not in stop_words]

    # Perform stemming and lemmatization
    words = [stemmer.stem(word) for word in words]
    words = [lemmatizer.lemmatize(word) for word in words]

    # Rejoin words into a single string
    cleaned_text = ' '.join(words)

    return cleaned_text

# Apply preprocessing to the dataset
df['cleaned_text'] = df['Text'].apply(preprocess_text)

df = df.drop(columns=['id', 'Tweet_ID', 'Username', 'Retweets', 'Likes'])
```

Then, the data is pre-processed. This process is done to clean the data, it includes removing stop words, stemming, lemmatizing, normalizing the words to lowercases, removing special characters and numbers, and also combining the cleaned text together. Once it is done, the columns which are id, Tweet_ID, Username, Retweets, Likes are removed to reduce redundancy when training the data later on.

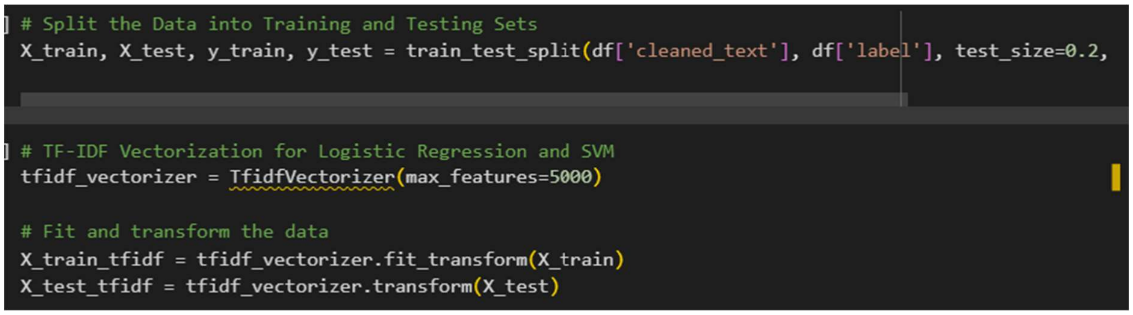


```
def get_sentiment_label(text):
    # Use TextBlob to calculate sentiment polarity
    blob = TextBlob(text)
    polarity = blob.sentiment.polarity

    # Assign sentiment based on polarity score
    if polarity > 0.1:
        return 2  # Positive
    elif polarity < -0.1:
        return 0  # Negative
    else:
        return 1  # Neutral

# Apply the sentiment labeling function
df['label'] = df['cleaned_text'].apply(get_sentiment_label)

# Display the first few rows to verify the labels
print(df.head())
```

Next, a Textblob is integrated into the code to calculate the sentiments so that the sentiments can be classified according to the polarity. The cleaned text will be applied with the Textblob and the dataset is checked to observed after the data is cleaned and applied with sentiment labels.

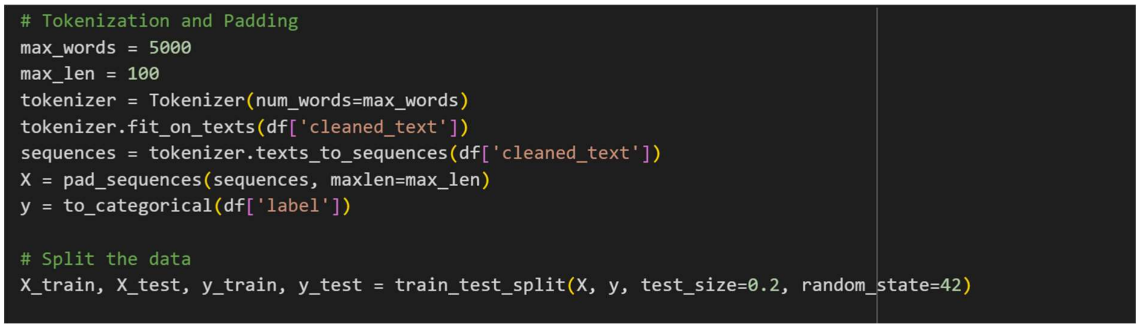


```
                                        cleaned_text  label
0  factor enjoy properti world main despit grow w...      1
1  televis sure other water hold valu economi men...      2
2  structur trip profession offer chair break cle...      0
3  walk method citi statement check analysi child...      1
4  singl bed cover nearli let feel side doctor ch...      2
```



```
# Split the Data into Training and Testing Sets
X_train, X_test, y_train, y_test = train_test_split(df['cleaned_text'], df['label'], test_size=0.2,

# TF-IDF Vectorization for Logistic Regression and SVM
tfidf_vectorizer = TfidfVectorizer(max_features=5000)

# Fit and transform the data
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

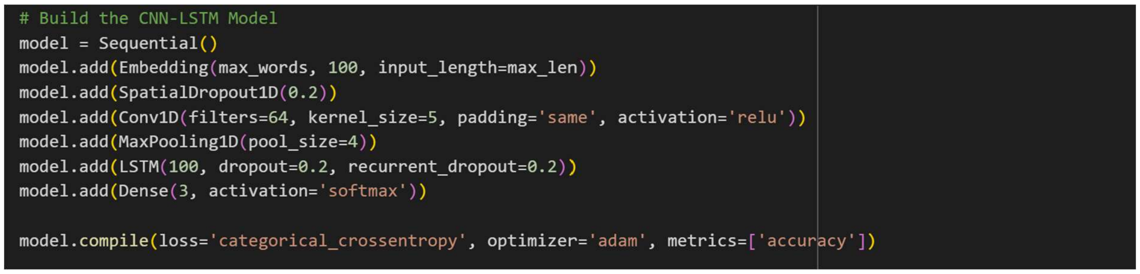
The dataset is then split into training and testing sets of test size 0.2 and random state at 42. After that, TF-IDF is implemented into logistic regression and SVM to convert text data to numerical form. The top 5000 words from the text data will be selected. These are the steps implemented before model and evaluation.



```
# Tokenization and Padding
max_words = 5000
max_len = 100
tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(df['cleaned_text'])
sequences = tokenizer.texts_to_sequences(df['cleaned_text'])
X = pad_sequences(sequences, maxlen=max_len)
y = to_categorical(df['label'])

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

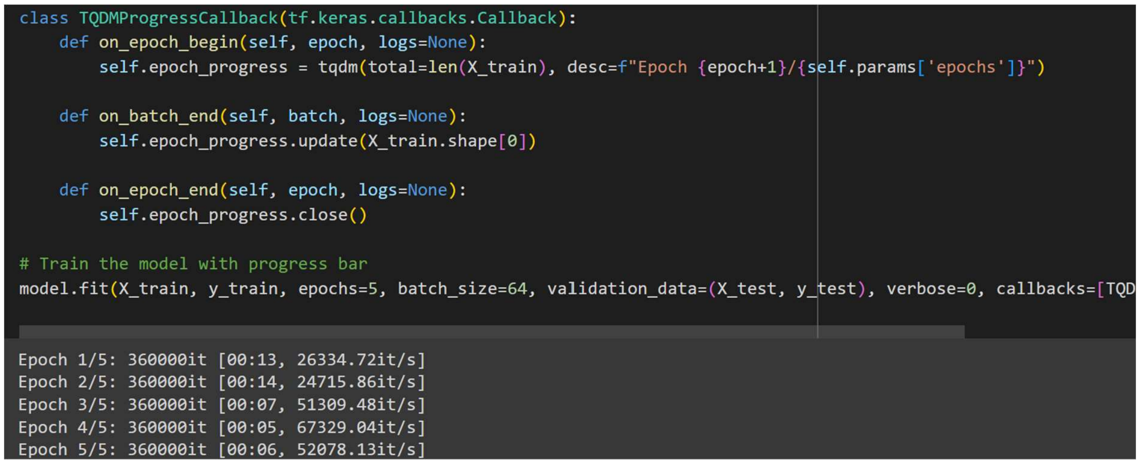
Next, another code is created for tokenizing the text data with the top 5000 words as well with the padding of length 100. This is to convert the labels into categorial for train test splits. This is done to prepare for CNN-LSTM hybrid model.

1. **This part shows the evaluation of the optimized techniques which are used in the hybrid model.**



```
# Build the CNN-LSTM Model
model = Sequential()
model.add(Embedding(max_words, 100, input_length=max_len))
model.add(SpatialDropout1D(0.2))
model.add(Conv1D(filters=64, kernel_size=5, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=4))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(3, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

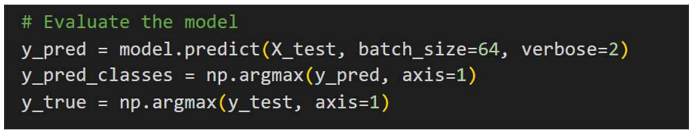
The CNN-LSTM model uses adam as the optimizer to stabilize and speed up the progress for achieving higher accuracy. The Categorical Cross-Entropy Loss function is used to measure the range between prediction and actual results. Whereas for the SpatialDropout, it is sued to prevent overftting in the model.



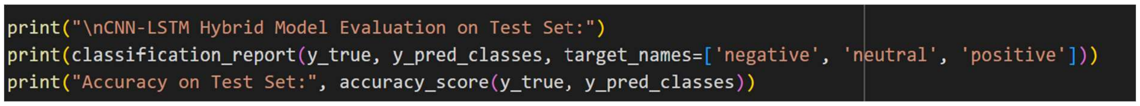
```
class TQDMProgressCallback(tf.keras.callbacks.Callback):
    def on_epoch_begin(self, epoch, logs=None):
        self.epoch_progress = tqdm(total=len(X_train), desc=f"Epoch {epoch+1}/{self.params['epochs']}")

    def on_batch_end(self, batch, logs=None):
        self.epoch_progress.update(X_train.shape[0])

    def on_epoch_end(self, epoch, logs=None):
        self.epoch_progress.close()

# Train the model with progress bar
model.fit(X_train, y_train, epochs=5, batch_size=64, validation_data=(X_test, y_test), verbose=0, callbacks=[TQD
```

```
Epoch 1/5: 360000it [00:13, 26334.72it/s]
Epoch 2/5: 360000it [00:14, 24715.86it/s]
Epoch 3/5: 360000it [00:07, 51309.48it/s]
Epoch 4/5: 360000it [00:05, 67329.04it/s]
Epoch 5/5: 360000it [00:06, 52078.13it/s]
```

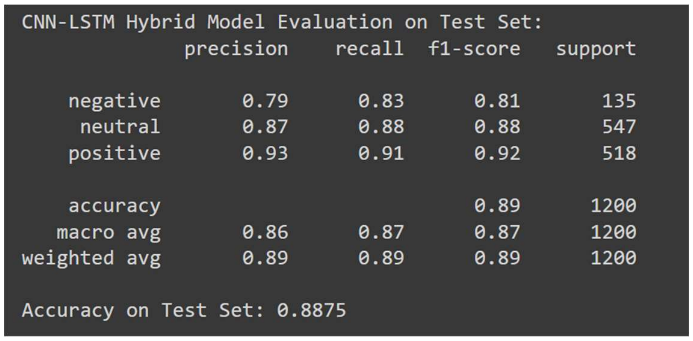
A progress bar callback is put here to monitor the real-time progress of the training. The model is trained over 5 epochs with a batch size of 64.



```
# Evaluate the model
y_pred = model.predict(X_test, batch_size=64, verbose=2)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)
```

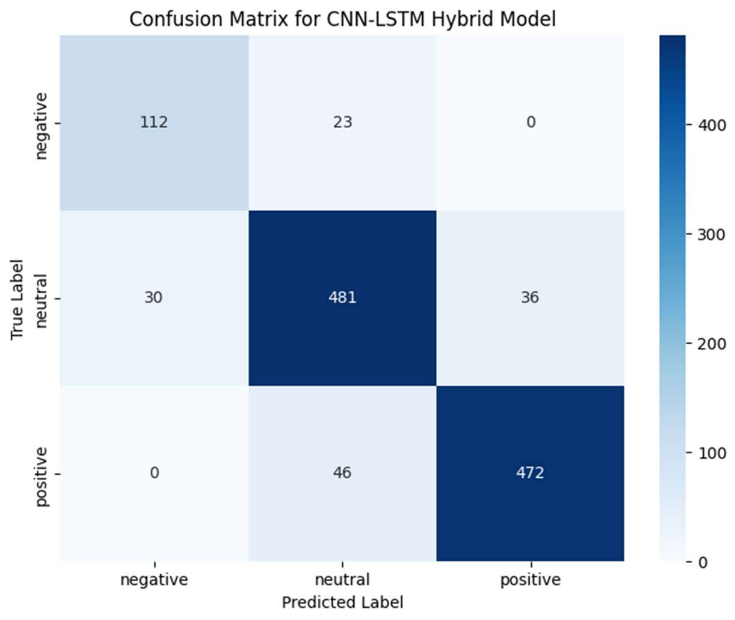


```
print("\nCNN-LSTM Hybrid Model Evaluation on Test Set:")
print(classification_report(y_true, y_pred_classes, target_names=['negative', 'neutral', 'positive']))
print("Accuracy on Test Set:", accuracy_score(y_true, y_pred_classes))
```



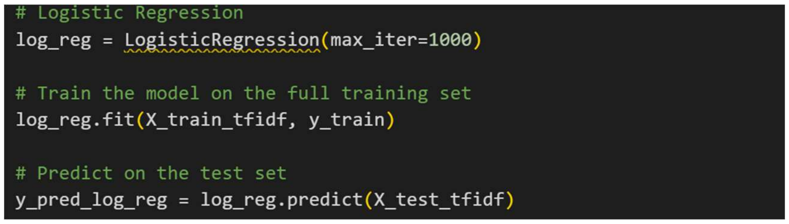
```
CNN-LSTM Hybrid Model Evaluation on Test Set:
              precision    recall  f1-score   support

    negative       0.79      0.83      0.81       135
     neutral       0.87      0.88      0.88       547
    positive       0.93      0.91      0.92       518

    accuracy                           0.89      1200
   macro avg       0.86      0.87      0.87      1200
weighted avg       0.89      0.89      0.89      1200

Accuracy on Test Set: 0.8875
```

Based on the results, CNN-LSTM model achieved an accuracy of 88.75%. In the “positive” class, it has the highest precision 0.93, recall 0.91, F1-score 0.92. The “neutral” class performed the best in recall and F1-score at 0.88, while “negative” class performed highest in recall at 0.83. The macro and weighted averages has displayed a balanced performance among the classes.



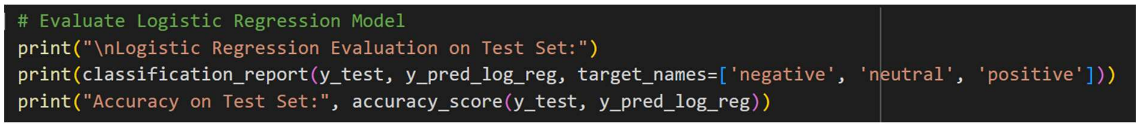
Based on the confusion matrix, the model displays the abilities to classify the three sentiments. The model shows high number of classifications correctly classify at neutral 481 and 472 for positive. This shows the model is learning the patterns effectively form the data.

2. This section will compare the techniques with Logistic regression and SVM.

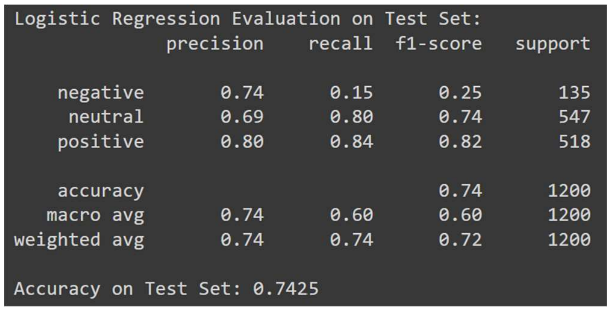


```
# Logistic Regression
log_reg = LogisticRegression(max_iter=1000)

# Train the model on the full training set
log_reg.fit(X_train_tfidf, y_train)

# Predict on the test set
y_pred_log_reg = log_reg.predict(X_test_tfidf)
```

In logistic regression, the optimization method used is to apply ‘max_iter=1000’ for sufficient iterations. The model is then train and test for prediction.

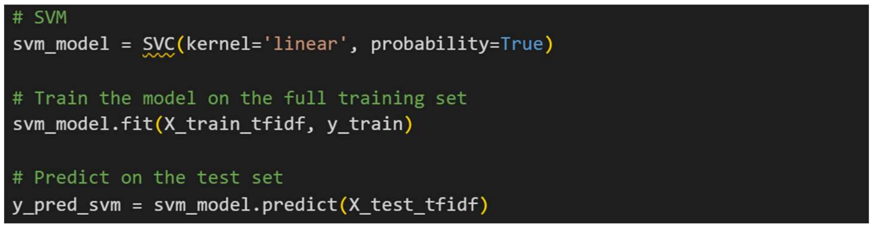


```
# Evaluate Logistic Regression Model
print("\nLogistic Regression Evaluation on Test Set:")
print(classification_report(y_test, y_pred_log_reg, target_names=['negative', 'neutral', 'positive']))
print("Accuracy on Test Set:", accuracy_score(y_test, y_pred_log_reg))
```



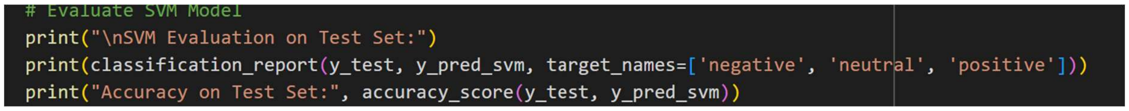
```
Logistic Regression Evaluation on Test Set:
              precision    recall  f1-score   support

    negative       0.74      0.15      0.25       135
     neutral       0.69      0.80      0.74       547
    positive       0.80      0.84      0.82       518

    accuracy                           0.74      1200
   macro avg       0.74      0.60      0.60      1200
weighted avg       0.74      0.74      0.72      1200

Accuracy on Test Set: 0.7425
```

Next, the data is evaluated. The logistic regression model obtained 74.25%. The “positive” class is highest at recall 0.84, “neutral” class highest is recall at 0.80 and “negative” class at precision 0.74, it performed poorly in recall and f1-score. “Positive” performs the strongest in the classes. The macro average shows moderate performance, weight average has more consistency.

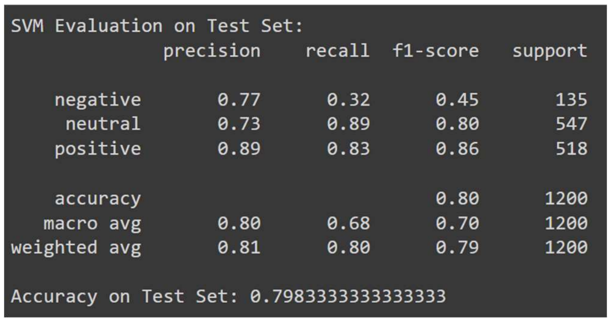


```
# SVM
svm_model = SVC(kernel='linear', probability=True)

# Train the model on the full training set
svm_model.fit(X_train_tfidf, y_train)

# Predict on the test set
y_pred_svm = svm_model.predict(X_test_tfidf)
```

SVM is created with the linear kernel, the optimization approach used here is to find the best margin which separates the different classes. The model is trained with TF-IDF and then it is put into prediction.



```
 # Evaluate SVM Model
 print("\nSVM Evaluation on Test Set:")
 print(classification_report(y_test, y_pred_svm, target_names=['negative', 'neutral', 'positive']))
 print("Accuracy on Test Set:", accuracy_score(y_test, y_pred_svm))
```



```
SVM Evaluation on Test Set:
              precision    recall  f1-score   support

    negative       0.77      0.32      0.45       135
     neutral       0.73      0.89      0.80       547
    positive       0.89      0.83      0.86       518

    accuracy                           0.80      1200
   macro avg       0.80      0.68      0.70      1200
weighted avg       0.81      0.80      0.79      1200

Accuracy on Test Set: 0.7983333333333333
```

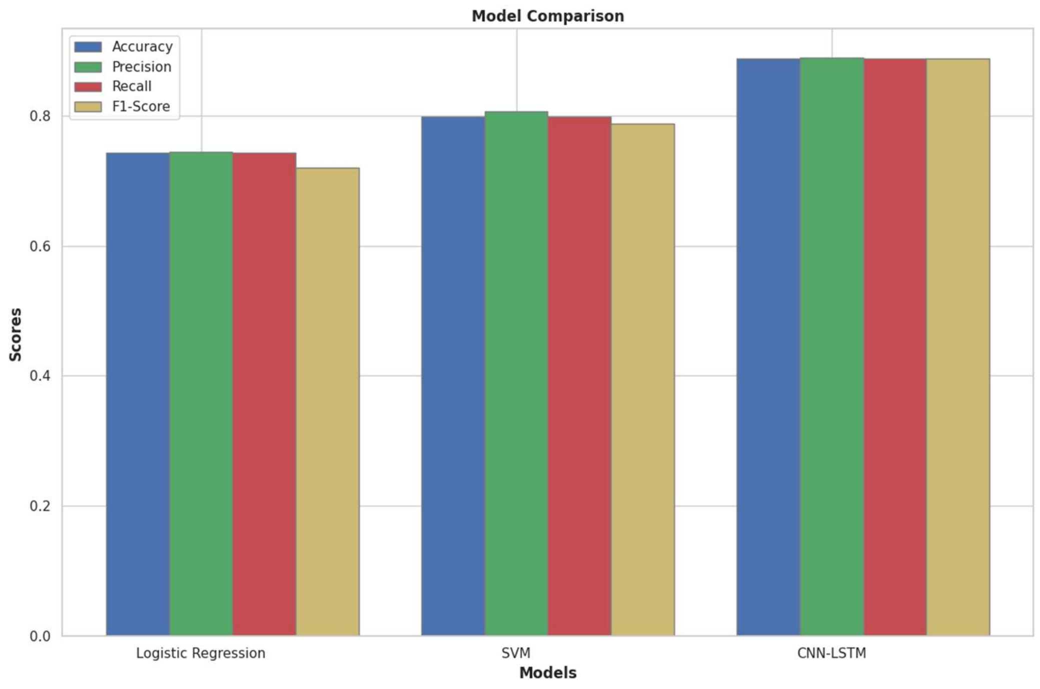
The SVM model has an accuracy of 79.83%. The "positive" class has the highest precision 0.89, "neutral" class has 0.89 for recall, "negative" class is highest in precision at 0.77. Both the averages of macro and weight has shown overall decent performance across the classes.

Comparing logistic regression and SVM with CNN-LSTM hybrid model, both logistic regression and SVM uses simple and fast methods which are easy implement while the CNN-LSTM model takes longer to process as it captures the patterns from the text data. For CNN-LSTM, the learning rate is more effective as it uses adam as the optimizer while Logistic regression and SVM relies on the pre-defined features.

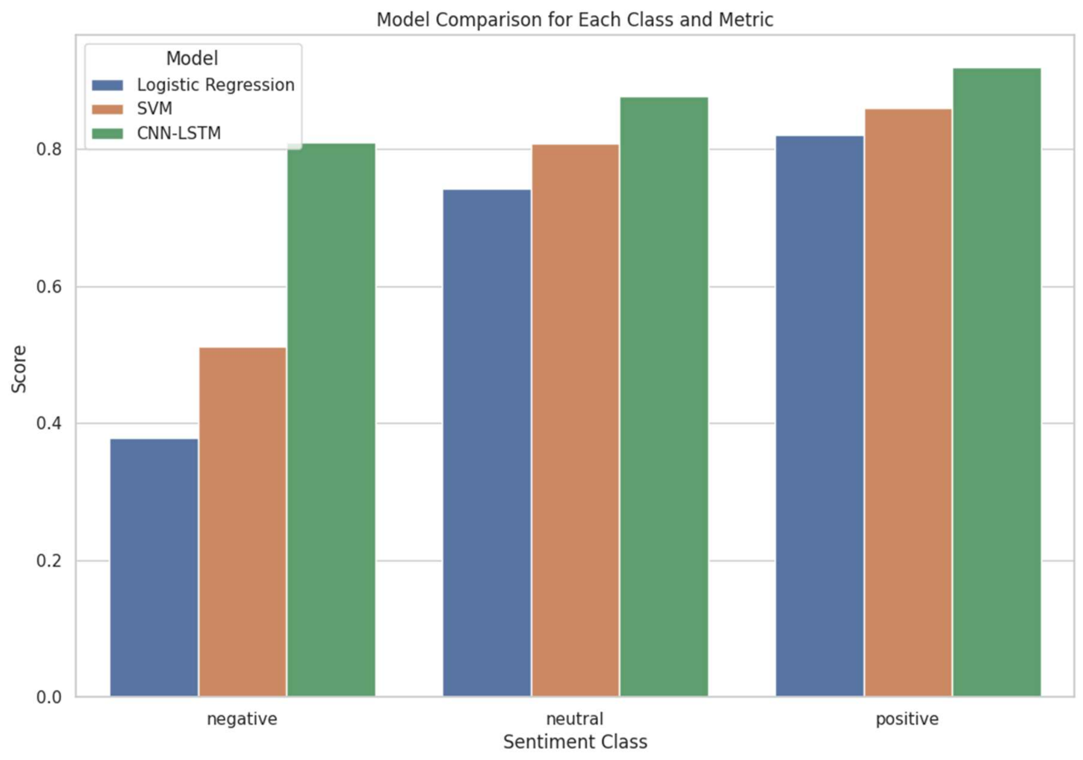
The below compares the performances across the models. It is observed that the CNN-LSTM hybrid model performs the best.

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.742500	0.744411	0.742500	0.719393
SVM	0.798333	0.805823	0.798333	0.787638
CNN-LSTM	0.887500	0.888455	0.887500	0.887879

This chart below shows the CNN-LSTM model outperforms the other models.



It is observed that based on the sentiment classes, CNN-LSTM Hybrid model performs the best as well.



3. **This section talks about the suitability for real-time sentiment analysis based on the models used in this study.**

The logistic regression model can be said to be a good choice for real-time sentiment analysis as it is simple and fast, therefore making it very effective for small, scaled datasets. However, it might have difficulty for complicated data. Whereas for SVM, the SVM performs well with large datasets in the case the dataset is separable in a straight line as the SVM is linear kernel. It might have difficulties as well if the data is really large sized and it is not separable. As for CNN-LSTM hybrid model, it can handle large complex datasets well which is suitable for text data from Twitter, with the combination of computational resources such as GPU, it can potentially speed up the process and improve its ability in real-time analysis.

In conclusion, based on comparing the optimization techniques, the CNN-LSTM hybrid model obtains high accuracy by using advanced techniques but these techniques require more computational resources. Logistic regression and SVM requires less, thus making it simpler and faster, therefore they are good options for running small datasets quickly. As for suitability in real-time analysis, the logistic regression and SVM are more suitable for real-time analysis if there are limited resources. If there is more computational resources, CNN-LSTM will be more suitable as it provides better accuracy and performance.

References

Behera, R. K., Jena, M., Rath, S. K., & Misra, S. (2021). Co-LSTM: Convolutional LSTM model for sentiment analysis in social big data. *Information Processing & Management*, *58*(1), 102435. https://doi.org/10.1016/j.ipm.2020.102435

Jain, P. K., Saravanan, V., & Pamula, R. (2021). A Hybrid CNN-LSTM: A Deep Learning Approach for Consumer Sentiment Analysis Using Qualitative User-Generated Contents. *ACM Transactions on Asian and Low-Resource Language Information Processing*, *20*(5), 1–15. https://doi.org/10.1145/3457206

Kamyab, M., Liu, G., & Adjeisah, M. (2021). Attention-Based CNN and Bi-LSTM Model Based on TF-IDF and GloVe Word Embedding for Sentiment Analysis. *Applied Sciences*, *11*(23), 11255. https://doi.org/10.3390/app112311255

Ronran, C., Lee, S., & Jang, H. J. (2020). Delayed Combination of Feature Embedding in Bidirectional LSTM CRF for NER. *Applied Sciences*, *10*(21), 7557. https://doi.org/10.3390/app10217557

Zhang, L., & Song, Q. (2022). Credit Evaluation of SMEs Based on GBDT-CNN-LR Hybrid Integrated Model. *Wireless Communications and Mobile Computing*, *2022*, 1–8. https://doi.org/10.1155/2022/5251228

Al-Abyadh, M. H. A., Iesa, M. a. M., Azeem, H. a. H. A., Singh, D. P., Kumar, P., Abdulamir, M., & Jalali, A. (2022). Deep Sentiment Analysis of Twitter Data Using a Hybrid Ghost Convolution Neural Network Model. *Computational Intelligence and Neuroscience*, *2022*, 1–8. https://doi.org/10.1155/2022/6595799